ML PROGRAMMER TO ML ARCHITECT

SKILLSOFT ASPIRE JOURNEY

Głównym wyzwaniem przed którym stają dziś organizacje na całym świecie jest konieczność ciągłego podnoszenia umiejętności i poziomu wiedzy w ślad za gwałtownym rozwojem nowych technologii i zmian na globalnym rynku.

Stały rozwój i podnoszenie kwalifikacji w IT od dawna jest już rzeczą oczywistą, a możliwość zapewnienia wsparcia specjalistom chcącym stale się rozwijać jest jedną z głównych kart przetargowych w walce o pracownika.

Na rynku liczą się dziś ludzie, którzy posiadają konkretne kompetencje i zestaw umiejętności pozwalający im wykonywać zadania efektywnie, a nie Ci z najdłuższym stażem pracy.

Dziś, bardziej niż kiedykolwiek w cenie jest umiejętność budowania ścieżki kariery dla profesjonalistów IT, którzy wciąż chcą się liczyć na rynku pracy.

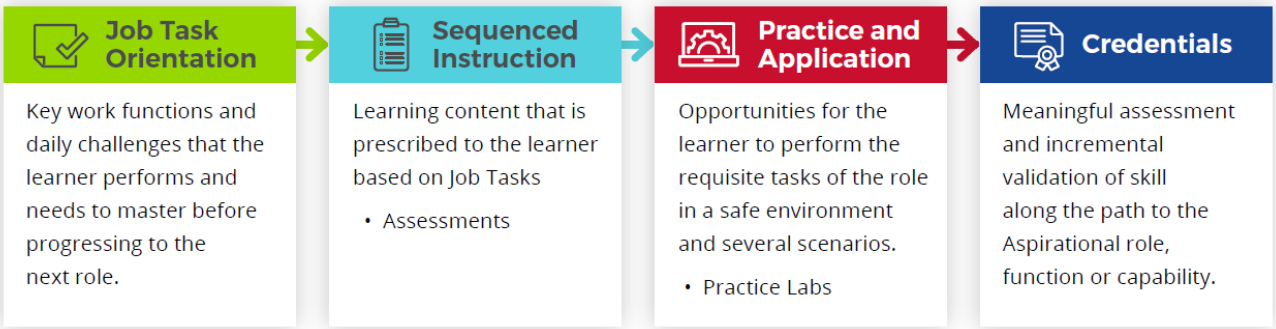
Skillsoft Aspire Journey stanowi odpowiedź na pytanie, jakie szkolenia muszę ukończyć, aby być przygotowanym do mojej wymarzonej pracy. Spośród kilkuset kanałów tematycznych dostępnych na naszej platformie szkoleniowej nasi specjaliści wybrali te, które naszym zdaniem najlepiej wyposażą uczących się w narzędzia potrzebne do realizacji zadań w nowej roli.

Skillsoft Aspire Journey to zestawy szkoleń i ćwiczeń w języku angielskim, które metodycznie, krok po kroku pozwalają specjalistom przejść od poziomu podstawowego do zaawansowanego.

Każda ścieżka zawiera szkolenia, laboratoria wirtualne, video i książki, które pomogą uczącym się osiągnąć pożądane kompetencje poświadczone certyfikatem.

Aspire Journey Model

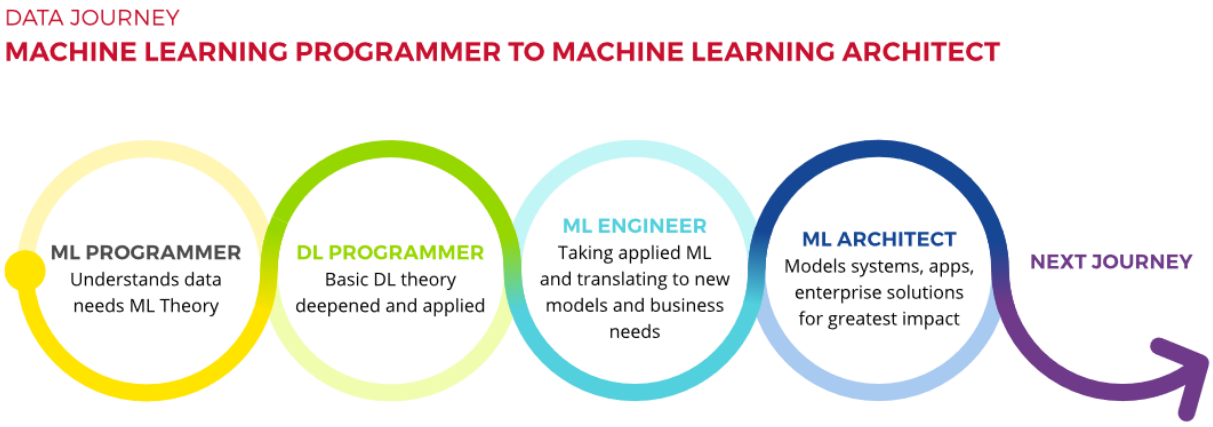
Cała ścieżka opiera się na 4-elementowym cyklu powtarzanym na kolejnych etapach nauki.



1. Określenie kluczowych funkcji i wyzwań, z którymi musi poradzić sobie uczący się w chwili obecnej, jak i tymi, z którymi przyjdzie mu się zmierzyć w nowej pracy.
2. Przejście zaprojektowanych ścieżek w proponowanej kolejności, wykonanie ćwiczeń i zaliczenie testów.
3. Przećwiczenie nowych umiejętności w kontrolowanym środowisku w oparciu o gotowe scenariusze działań. Laboratoria wirtualne Skillsoft
4. Certyfikat – zaliczenie testu końcowego na poziomie co najmniej 70% i uzyskanie certyfikatu potwierdzającego ukończenie danego etapu nauki.

Aspire Journey – ML Programmer to ML Architect

Analizując trendy opisujące zachowanie użytkowników na naszych platformach szkoleniowych i współpracując ściśle z naszymi klientami na całym świecie Skillsoft wyselekcjonował najlepsze materiały szkoleniowe i ułożył je w ustrukturalizowaną ścieżkę rozwoju. Ścieżka zawiera ponad 72 godzin szkoleniowych.



24 courses
22h 5m 44s

- linear regression
- computational theory
- training sets

21 courses

21h 30m 28s

- neural networks
- CNNs, RNNs
- ML algorithms

16 courses
16h 22m 55s

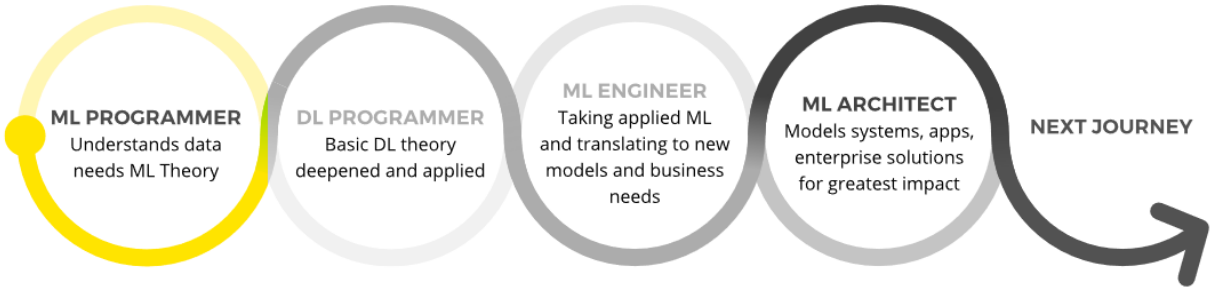
- predictive modeling and analytics
- ML modeling
- ML architecting.

11 courses
12h 27m 38s

- applied predictive modeling
- CNNs and RNNs, ML algorithms.

DATA JOURNEY

MACHINE LEARNING PROGRAMMER TO MACHINE LEARNING ARCHITECT



Track 1: ML Programmer (duration: 22h 5m 44s)

 NLP for ML with Python: NLP Using Python & NLTK	 NLP for ML with Python: Advanced NLP Using spaCy & Scikit-learn
Objectives: ▪ discover the key concepts covered in this course ▪ define NLP, it uses, and the benefits and challenges associated with it ▪ recall essential NLP terms and the steps involved in natural language processing ▪ describe the rule-based and probabilistic parsing approaches and the different types of parsers that are used in NLP ▪ define corpus and corpus linguistic and describe the benefits associated with corpus linguistic ▪ implement regular expressions in Python ▪ list prominent Python NLP libraries and their capabilities ▪ set up and configure the NLTK environment to illustrate how to process raw texts ▪ recognize the major components of NLP ▪ define tokenization and illustrate different tokenization use cases with NLTK ▪ demonstrate various tokenization use cases with NLTK ▪ filter stop words in a tokenized sentence using NLTK ▪ list NLP terminologies, recall Python NLP libraries, and filter stop words in a tokenized sentence using NLTK	Objectives: ▪ discover the key concepts covered in this course ▪ demonstrate stemming and lemmatization scenarios in NLP using NLTK ▪ extract synonyms and antonyms from NLTK WordNet using Python ▪ demonstrate the steps involved in extracting topics using LDA ▪ describe NER, its use cases, and the standard libraries that use NER ▪ describe the concept of POS tagging, its importance in the context of NLP and the various implementations in NLTK ▪ recognize the essential features provided by spaCy for NLP ▪ analyze and process texts using spaCy ▪ implement TF and TF-IDF text classification using Python, scikit-learn, and NLTK ▪ implement sentiment analysis using Python and scikit-learn ▪ recall the differences between stemming and lemmatization, list the prominent features of spaCy, and implement sentiment analysis using Python and scikit-learn

<table>
<tr>
<td> 
</td>
<td>Linear Algebra and Probability: Fundamentals of Linear Algebra</td>
<td> 
</td>
<td>Linear Algebra & Probability: Advanced Linear Algebra</td>
</tr>
<tr>
<td colspan="2">Objectives:

▪ discover the key concepts covered in this course
▪ identify the essential characteristics of linear algebra and its role in machine learning implementations
▪ list the important classes of spaces associated with linear algebra
▪ describe features of vector spaces and list the different types of vector spaces and their application in distribution and Fourier analysis
▪ describe the concept of inner product spaces and the various theorems that are applied on inner product spaces
▪ demonstrate how to implement vector arithmetic using Python
▪ demonstrate how to implement vector scalar multiplication using Python
▪ describe the concept and different types of vector norms
▪ implement matrix-matrix multiplication, matrix-vector multiplication, and matric-scalar multiplication using Python
▪ recognize operations that can be performed on matrix, such as matrix norms and matrix identities
▪ recognize how the trace, determinant, inverse, and transpose operations are applied on matrix
▪ describe matrix decomposition, using eigendecomposition, and the role of Eigenvectors and Eigenvalues
▪ describe the features of vector spaces, recall the different types of vector norms, and implement matrix-matrix multiplication, matrix-vector multiplication, and matric-scalar multiplication using Python</td>
<td colspan="2">Objectives:

▪ discover the key concepts covered in this course
▪ use Python libraries to implement principal component analysis with matrix multiplication
▪ describe sparse matrix and the operations that can be performed on sparse matrix
▪ define the concept of tensors in linear algebra and list the arithmetic operations that can be applied on tensors
▪ implement Hadamard product on tensors using Python
▪ describe singular-value decomposition and how to calculate it
▪ reconstruct a rectangular matrix from single-value decomposition
▪ recognize the characteristics of probability that are applicable in machine learning
▪ describe probability in linear algebra and its role in machine learning
▪ recall the types of random variables and the functions that can be used to manage random numbers in probability
▪ describe the concept and characteristics of central limit theorem and means and recognize common usage scenarios
▪ describe parameter estimation and distribution using Gaussian
▪ describe binomial distribution and its characteristics
▪ recall the arithmetic operations that can be applied on tensors, list the features of multivariate statistics that are applicable in machine learning, and implement Hadamard product on tensors using Python</td>
</tr>
</table>

Linear Regression Models: Introduction

Objectives:

- define what regression is and recall how it can be used to represent a relationship between two variables
- identify the applications of regression and recognize why it is used to make predictions
- describe how to evaluate the quality of a regression model by measuring its loss
- recognize the specific relationship which needs to exist between the input and output of a regression model
- describe the technique used in order to make predictions with regression models
- compare classic ML and deep learning techniques to perform a regression
- identify the various components of a neural network such as neurons and layers and how they fit together
- recall the two types of functions used in a neuron and their individual roles
- describe the configurations required to use a neuron for linear regression
- list the steps involved in calculating the optimal weights and biases of a neural network
- define the technique of gradient descent optimization in order to find the optimal parameters for a neural network
- recall key concepts of linear regression and deep learning

Linear Regression Models: Building Models with Scikit & Keras

Objectives:

- use the Pandas library to load a dataset in the form of a CSV file into a Dataframe for consumption by a linear regression model
- create training and validation sets for your regression model
- configure a linear regression model and then train and validate it and view the metrics for the model and visualize it using Matplotlib
- install the Keras library and prepare the dataset for consumption by a Keras model
- define the architecture for a Keras sequential model and initialize it
- compile a Keras sequential model by defining the loss function and optimizer and train it to get the optimal values for weights and biases
- evaluate a Keras sequential model by using it to make predictions on test data
- work with training sets and the Keras sequential model

Linear Regression Models: Multiple & Parsimonious

Objectives:

- identify the reasons to use multiple features when doing a regression and the technique involved in creating such a multiple regression model
- prepare a dataset containing multiple features to used for training and evaluating a linear regression model
- configure, train and evaluate the linear regression model which makes predictions from multiple input features
- create a dataset with multiple features in a form which can be fed to a neural network for training and validation
- define the architecture for a Keras sequential model and set the training parameters such as loss function and optimizer
- make predictions on the test data and examine the metrics to gauge the quality of the neural network model
- use Pandas and Seaborn to visualize correlations in a dataset and identify features which convey similar information
- identify the risks involved with multiple regression and the need to select features carefully
- apply the principle of parsimonious regression to re-build the Linear Regression model and compare the results with the kitchen sink approach
- build a Keras model after selecting only the important features from a dataset
- encode categorical integers for ML algorithms as well as use Pandas and Seaborn to view correlations, and enumerate risks

Linear Regression Models: Introduction to Logistic

Objectives:

- identify the types of problems which can be solved by logistic regression
- describe the qualities of a logistic regression S-curve and understand the kind of data it can model
- recognize how a logistic regression can be used to perform classification tasks
- compare logistic regression with linear regression
- recall how neural networks can be used to perform a logistic regression
- prepare a dataset to build, train and evaluate a logistic regression model in Scikit Learn
- use a logistic regression model to perform a classification task and evaluate the performance of the model
- prepare a dataset to build, train and evaluate a Keras sequential model
- build, train and validate the Keras model by defining various components including the activation functions, optimizers and the loss function
- employ key classification techniques in logistical regression

 Simplifying Regression and Classification with Estimators	 Computational Theory: Language Principle & Finite Automata Theory
Objectives: ▪ describe the role of estimators in speeding up the development of standard regression and classification models ▪ prepare a dataset to be used to train and validate a linear regression estimator ▪ use the estimator's methods to train and evaluate the model and visualize its performance using Matplotlib ▪ transform a dataset so that it can be used to train and validate a linear classifier estimator ▪ use input functions to pass training and validation data to an estimator and evaluate its performance on ▪ utilize TensorFlow estimators with linera regression models	Objectives: ▪ define the theory of computation and list the prominent branches of computation ▪ list the prominent models of computation ▪ specify the concept of automata theory and list the prominent classes of automata ▪ define the principles of finite state machine ▪ recognize the essential principles driving formal language and automata theory ▪ recall the essential elements of the theory of formal language ▪ define regular expressions and list the theorems that are used to manage the semantics of regular expressions ▪ define regular grammar and list the essential grammars that are used to generate regular languages ▪ list the essential closure properties and theorems associated with regular language ▪ define context-free grammar and list its prominent features ▪ identify practical usage, branches, and models of computational theory, specify notations of formal language, and list types of context-free grammar

 Computational Theory: Using Turing, Transducers, & Complexity Classes	 Model Management: Building Machine Learning Models & Pipelines
Objectives: ▪ recall the analytical capabilities of grammar ▪ use Chomsky normal forms and Greibach normal forms to manage context-free grammars ▪ describe pushdown automata and the features of non-deterministic pushdown automata ▪ describe Turing machines and their capabilities ▪ list the prominent variations of themes that can be used to build Turing machines ▪ describe finite transducers and list the prominent types ▪ recall the underlying limitations of algorithmic computations ▪ recognize how computational complexities can impact Turing machine models and language families ▪ recall how to manage computational complexities using P and NP classes ▪ define recursive and recursively enumerable languages and their essential properties ▪ describe properties and variations of Turing machines, types of finite transducers, and properties of recursively enumerable language	Objectives: ▪ recognize the differences between machine learning models and algorithms ▪ identify the different types of machine learning models ▪ describe the approaches and steps involved in developing machine learning models ▪ create and save machine learning models using scikit-learn ▪ list machine learning models that can be used to manage classification and regression problems ▪ build machine learning pipelines ▪ list prominent tools that can be used to build machine learning pipelines ▪ implement machine learning pipelines using scikit-learn ▪ recall the steps involved in iterative machine learning model management and the associated benefits ▪ build machine learning models and pipelines using scikit-learn

<table>
<tr>
<td> 

Model Management: Building & Deploying Machine Learning Models in Production</td>
<td> 

Bayesian Methods: Bayesian Concepts & Core Components</td>
</tr>
<tr>
<td>Objectives:

▪ describe hyperparameter and the different types of hyperparameter tuning methods
▪ demonstrate how to tune hyperparameters using grid search
▪ recognize the essential aspects of a reproducible study
▪ list machine learning metrics that can be used to evaluate machine learning algorithms
▪ recognize the relevance of versioning machine learning models
▪ implement version control for machine learning models using Git and DVC
▪ describe the architecture of ModelDB used for managing machine learning models
▪ list essential features of the model management framework
▪ set up Studio.ml to manage machine learning models
▪ create machine learning models in production
▪ set up machine learning models in production using Flask
▪ deploy machine or deep learning models in production
▪ tune hyperparameter with grid search, version machine learning model using Git, and create machine learning models for production</td>
<td>Objectives:

▪ discover the key concepts covered in this course
▪ describe the concept of Bayesian probability and statistical inference
▪ describe the concept of Bayes' theorem and its implementation in machine learning
▪ identify the role of probability and statistics in Bayesian analysis from the perspective of frequentist and subjective probability paradigms
▪ describe standard probability, continuous, and discrete distribution
▪ recall the essential ingredients of Bayesian statistics including prior distribution, likelihood function, and posterior inference
▪ recognize the implementation of prominent Bayesian methods including inference, statistical modeling, influence of prior belief, and statistical graphics
▪ identify the core concepts of Bayesian machine learning from the perspective of modeling, sampling algorithms, and variation inference
▪ describe prior knowledge and compare the differences between non-informative prior distribution and informative prior distribution
▪ recall the steps involved in Bayesian analysis, including modeling data, deciding prior distribution, likelihood construction, and posterior distribution
▪ specify the essential ingredients of Bayesian statistics and recall the prominent Bayesian methods and the steps involved in Bayesian analysis</td>
</tr>
</table>

<table>
<tr>
<td> 

Implementing Bayesian Model and Computation with PyMC</td>
<td> 

Bayesian Methods: Advanced Bayesian Computation Model</td>
</tr>
<tr>
<td>Objectives:

▪ discover the key concepts covered in this course
▪ identify critical features of and the difficulties associated with Bayesian learning methods
▪ define the Bayesian model and classify single-parameter, multi-parameter, and hierarchical Bayesian models
▪ describe the features of probabilistic programming and list the popular probabilistic programming languages
▪ use PyMC to define a model and arbitrary deterministic function and use the model to generate posterior samples
▪ recall the fundamental activities involved in Bayesian data analysis process, including model checking, evaluation, comparison, and model expansion
▪ implement Bayesian data analysis with PyMC using the rejection sampling approach
▪ recognize the essential approaches that can be used to implement Bayesian computation, including numerical integration, distributional approximation, and direct simulation
▪ describe Markov chain simulation and how it is used for computations
▪ implement Markov chain simulation using Python
▪ list the prominent algorithms that can be used to find posterior modes based on the distribution approximation
▪ specify the essential features of probabilistic programming, recall the approaches that can be used to implement Bayesian computation, and implement Bayesian data analysis using PyMC</td>
<td>Objectives:

▪ discover the key concepts covered in this course
▪ demonstrate how to build and implement Bayesian linear regression models using Python
▪ list the prominent hierarchical linear models from the perspective of regression coefficients
▪ describe the concept of probability models and illustrate the use of Bayesian methods for problems with missing data
▪ demonstrate how to build probability models using Python
▪ describe non-linear and non-parametric models from the perspective of coefficient shrinkage and multivariate regression
▪ specify the fundamental concepts of Gaussian process models
▪ recognize the approaches of using mixture models for classification and regression
▪ define and list the essential properties of Dirichlet process models
▪ demonstrate how to implement Bayesian inference models in Python with PyMC3
▪ recall hierarchical linear models from the perspective of regression coefficients, describe the approach of working with generalized linear models, and implement Bayesian inference using PyMC3
▪</td>
</tr>
</table>

 Reinforcement Learning: Essentials	 Reinforcement Learning: Tools & Frameworks
Objectives: ▪ define reinforcement learning and describe its essential elements ▪ recognize the key differences between the reinforcement learning and machine learning paradigms ▪ depict the flow of reinforcement learning using agent, action, and environment ▪ describe different state change scenarios and transition processes in reinforcement learning ▪ recognize the role of rewards in reinforcement learning ▪ list the essential steps agents take to make decisions in reinforcement learning ▪ recognize prominent reinforcement learning environment types ▪ install OpenAI Gym and OpenAI Universe ▪ list reinforcement learning elements, agents involved in the process and the steps they take, and reinforcement learning environments	Objectives: ▪ recognize the different types of reinforcement learning that can be implemented for decision-making ▪ implement reinforcement learning using Keras and Python ▪ identify the role of the Markov decision process in reinforcement learning ▪ describe Q-learning, Q-learning rule, and deep Q-learning ▪ install TensorFlow ▪ implement reinforcement learning using TensorFlow ▪ implement Q-learning using Python ▪ implement reinforcement learning using Python and TensorFlow and implement Q-learning using Python

<table>
<tr>
<td> 

Math for Data Science & Machine Learning</td>
<td> 

Building ML Training Sets: Introduction</td>
</tr>
<tr>
<td>Objectives:

▪ understand how to work with vectors in Python
▪ understand basis and projection of vectors in Python
▪ understand how to work with matrices in Python
▪ understand how to multiply matrices in Python
▪ understand how to divide matrices in Python
▪ understand how to work with linear transformations in Python
▪ understand how to apply gaussian elimination in Python
▪ understand how to work with determinants in Python
▪ understand how to work with orthogonal matrices in Python
▪ recognize how to obtain eigenvalues from eigen decomposition in Python
▪ recognize how to obtain eigenvectors from eigen decomposition in Python
▪ recognize how to obtain pseudo inverse in Python
▪ work with math for data science and machine learning</td>
<td>Objectives:

▪ use the Pandas library to load a dataset in the form of a CSV file and perform some exploratory analysis on its features
▪ transform the continuous data in a series to binary values by using scikit-learn's Binarizer
▪ apply the MinMaxScaler on a dataset to get two similar columns to have the same range of values
▪ standardize multiple columns in your dataset using scikit-learn's StandardScaler
▪ distinguish between the Normalizer and other scaling techniques and apply this scaler on the continuous features of a dataset
▪ represent the values in a column as a proportion of the maximum absolute value by using the MaxAbsScaler
▪ apply label encoding on the features and target in your dataset and recognize its limitations when applied on input features
▪ use the Pandas library to one-hot encode one or more features of your dataset and distinguish between this technique and label encoding
▪ transform a continuous series into a categorical (binary) one, distinguish between Normalization and other scaling techniques, score each product as a proportion of the top product’s sales, and encode the ”VehicleType” field which contains values [“Hatchback”, “Sedan”, “SUV”]</td>
</tr>
</table>

<table>
<tr>
<td> 

Building ML Training Sets: Preprocessing Datasets for Linear Regression</td>
<td> 

Building ML Training Sets: Preprocessing Datasets for Classification</td>
</tr>
<tr>
<td>Objectives:

▪ use the Pandas library to load a csv file into a dataframe and analyze its contents using Pandas and Matplotlib
▪ create a linear regression model using scikit-learn to predict the sale price of a house and evaluate this model using metrics such as mean squared error and r-square
▪ apply min-max scaling on the continuous fields and one-hot encoding on the categorical columns of a dataset
▪ recognize the benefits of scaling and encoding datasets by evaluating the performance of a regression model built with preprocessed data
▪ use scikit-learn's StandardScaler on the continuous features of a dataset and compare its effects with that of min-max scaling
▪ identify the characteristics of the StandardScaler, encode a feature column which contains certain values, recall two metrics used to evaluate regression models, and enumerate the details conveyed in a Boxplot</td>
<td>Objectives:

▪ use the Pandas library to load a CSV dataset into a dataframe and scale the continuous features using a standard scaler
▪ build and evaluate a support vector classifier in scikit-learn, use Pandas and Seaborn to generate a heatmap, and spot the correlations between features in a dataset
▪ apply the technique of Principal Component Analysis to reduce the number of dimensions in your input data and obtain the explained variance of each principal component
▪ apply normalization and PCA on a dataset and build a classification model with the principal components of scaled data
▪ encode the target column of a dataset containing certain values, identify the features of Normalization, enumerate reasons for using PCA, split data into training and test sets using scikit-learn, identify one method of viewing correlations in a dataset using Pandas and Seaborn</td>
</tr>
</table>

Linear Models & Gradient Descent: Managing Linear Models

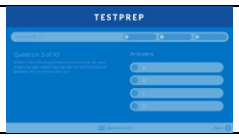
Objectives:

- discover the key concepts covered in this course
- define linear model and the various classification of linear models that are used in predictive analytics
- recognize the different statistical approaches that are used to implement linear models (single regression, multiple regression and ANOVA)
- define generalized linear model and the various essential components of generalized linear model (random component, linear predictor and link function)
- compare the differences between the ANOVA and ANCOVA approaches of statistical test
- demonstrate the implementation of linear regression models using Scikit-learn
- describe the concept of bias, variance and regularization and their usages in evaluating predictive models
- define the concept of ensemble techniques and illustrate how bagging and boosting algorithms are used to manage predictions
- implement bagging algorithms with the approach of random forest using Scikit-learn
- implement boosting ensemble algorithms using Adaboost classifier in Python
- list the classifications of linear models, recall the essential components of generalized linear models, and implement boosting algorithm using Adaboost classifier

Linear Models & Gradient Descent: Gradient Descent and Regularization

- Objectives:
-
- discover the key concepts covered in this course
- list and describe the characteristics of the prominent types of linear regression
- describe the essential features of simple and multiple regressions and how they're used to implement linear models
- demonstrate how to implement simple regression models using Python libraries
- implement multiple regression models in Python using Scikit-learn and StatsModels
- define gradient descent and the different types of gradient descent
- classify the prominent gradient descent optimization algorithms from the perspective of their mathematical representation
- implement a simple representation of gradient descent using Python
- implement linear regression using mini-batch gradient descent to compute hypothesis and predictions
- describe the benefits of regularization and the objective of L1 & L2 regularization
- demonstrate how to implement L1 and L2 regularization of linear models using Scikit-learn
- recall the essential features of simple and multiple regression, implement a simple regression model using Python and implement L1 regularization using Scikit-learn

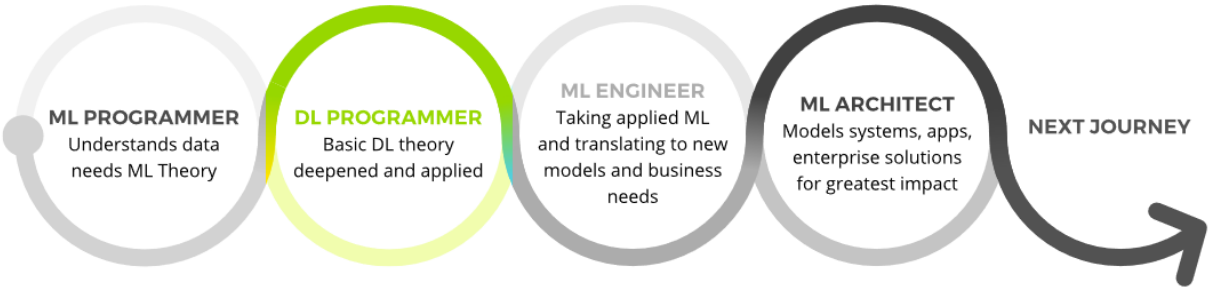
LAB: ML Programmer Practice Lab



Final Exam: ML Programmer TestPrep

DATA JOURNEY

MACHINE LEARNING PROGRAMMER TO MACHINE LEARNING ARCHITECT



Track 2: DL Programmer (duration: 21h 30m 28s)

	
Objectives ▪ discover the key concepts covered in this course ▪ identify the characteristics of biological neural networks that inspired artificial neural networks ▪ list the essential components of biological neural networks and describe the signal processing mechanism of biological neural networks ▪ describe essential components of artificial neural networks and their capabilities ▪ recognize layered architectural patterns that can be used to implement neural networks ▪ classify the various computational models that can be implemented using the neural networks paradigm ▪ describe the interconnection between neurons involving weights and fixed weights ▪ describe threshold functions and the basic logic gates of AND, OR, and XNOR ▪ implement neural networks using Python and the core libraries provided by Python for neural networks ▪ create a neural network model using Python, Keras, and TensorFlow ▪ list prominent use cases of implementing neural networks ▪ recall the essential components of artificial neural networks, list the prominent use cases of neural networks, and implement neural networks using Python	▪ Objectives ▪ ▪ discover the key concepts covered in this course ▪ describe perceptrons and the computational role they play in artificial neural networks ▪ recognize the algorithms that can be used to implement single layer perceptron training models ▪ define multilayer perceptrons and illustrate the algorithmic difference from single layer perceptrons ▪ classify the role of linear and non-linear functions in perceptrons ▪ demonstrate the implementation of perceptrons using Python ▪ describe approaches and benefits of using the backpropagation algorithm in neural networks ▪ recognize the uses of linear and non-linear activation functions in artificial neural networks ▪ implement a simple perceptron classifier using Python ▪ recall the benefits of using the backpropagation algorithm in neural networks, and implement perceptrons and perceptron classifiers using Python ▪ ▪

	 Building Neural Networks: Artificial Neural Networks Using Frameworks Neural Networks
Objectives ▪ identify the key subject areas covered in this course ▪ describe the essential artificial neural network components that are required for processing data ▪ recognize the different paradigms of learning that are used in artificial neural network ▪ list the essential clustering techniques that can be applied on artificial neural network ▪ recognize the roles of the essential components that are used in building neural networks ▪ recall the approach of generating deep neural networks from perceptrons ▪ classify the differences between models and hyperparameter and specify the approach of tuning hyperparameters ▪ define the prominent types of classification algorithm that can be used in neural networks ▪ describe the prominent features of essential deep learning frameworks for building neural networks ▪ recognize how to choose the right neural network framework for neural network implementations from the perspective of usage scenarios and fitment model ▪ define the computational models that can be used to build neural network models ▪ list the essential components of ANN for processing data, recall the clustering techniques that can be applied on ANN, differentiate between models and hyperparameters, and specify the types of classification algorithms that can be used in ANN	Objectives ▪ identify the key subject areas covered in this course ▪ list the prominent building blocks involved in building a neural network ▪ recall the concept and characteristics of evolutionary algorithms, gradient descent, and genetic algorithms ▪ build neural networks using Python and Keras for classification with Tensorflow as the backend ▪ build neural networks using PyTorch ▪ implement object image classification using neural network algorithms ▪ define and illustrate the use of learning rates to optimize deep learning ▪ describe the various parameters and approaches of optimizing neural network speed ▪ demonstrate how to select hyperparameters and tune for dense networks using Hyperas ▪ build linear models with estimators using the capabilities of TensorFlow ▪ specify approaches that can be used to implement predictions with neural networks ▪ describe the temporal and heterogenous approaches of optimizing predictions ▪ build a neural network using Python and Keras, tune dense networks using Hyperas, and build a linear model with TensorFlow

<table>
<tr>
<td> 

Training Neural Networks: Implementing the Learning Process</td>
<td> 

Training Neural Networks: Advanced Learning Algorithms</td>
</tr>
<tr>
<td>Objectives

▪ identify the subject areas covered in this course
▪ describe the characteristics of perceptrons and neural networks
▪ recognize the essential components of perceptrons and perceptron learning algorithms
▪ identify the different types of learning rules that can be applied in neural networks
▪ compare the supervised and unsupervised learning methods of artificial neural networks
▪ list neural network algorithms that can be used to solve complex problems across domains
▪ prepare and curate data for neural network training implementation
▪ implement the artificial neural network training process using Python
▪ recall the algorithms that can be used to train neural networks
▪ implement backpropagation using Python to train artificial neural networks
▪ use backpropagation and Keras to implement multi-layer perceptron or neural net
▪ implement regularization in multilayer perceptron using Keras
▪ compare the supervised and unsupervised learning methods, recall algorithms that can be used to train neural networks, and implement backpropagation using Python to train ANN</td>
<td>Objectives

▪ identify the subject areas covered in this course
▪ describe features of online and offline training methods in artificial neural network
▪ describe the training patterns and teaching inputs that are used in artificial neural networks
▪ describe the approach of managing training samples
▪ implement overfitting and underfitting using baseline model
▪ describe the regularization techniques used in deep neural network
▪ train built models of neural networks using Python to implement prediction with high accuracy
▪ list the prominent training algorithms that are used for pattern association
▪ describe the architecture along with the algorithm associated with learning vector quantization
▪ define the essential phases involved in implementing Hebbian learning
▪ implement the Hebbian learning rule using Python
▪ describe the steps involved in implementing competitive learning
▪ list approaches of optimizing neural networks
▪ debug neural networks
▪ recall the training algorithms used for pattern association, list the steps of implementing competitive learning, and implement the Hebbian learning rule using Python</td>
</tr>
</table>

Improving Neural Networks: Neural Network Performance Management Neural Networks

Objectives

- discover the key concepts covered in this course
- describe the iterative workflow for machine learning problems with focus on essential measures and evaluation protocols
- recognize the machine learning problems that can be addressed using hyperparameters along with the various hyperparameter tuning methods and the problems associated with hyperparameter optimization
- recall the steps to improve the performances of neural networks along with impact of dataset sizes on neural network models and performance estimates
- demonstrate the impact of the size of training dataset on the quality of mapping function and the estimated performance of a fit neural network model
- recall the approaches of identifying overfitting scenarios and preventing overfitting using regularization techniques
- recognize the critical problems associated with neural networks along with the essential approaches of resolving them
- describe the impact of bias and variances on machine learning algorithms and recall the approaches of fixing high bias and high variance in data sets
- demonstrate how to trade off bias variance by building and deriving an ideal learning curve using Python
- recognize the various approaches of improving the performance of machine learning using data, algorithm, algorithm tuning and ensembles
- demonstrate how to test multiple models and select the right model using Scikit-learn
- specify the machine learning problems that we can address using hyperparameters, describe the impact of bias and variances on machine learning algorithms and test multiple models using Scikit-learn

Improving Neural Networks: Loss Function & Optimization Neural Networks

Objectives

- discover the key concepts covered in this course
- define Loss function and list the various types of loss function
- recognize the impacts of the different types of loss function on neural networks models
- calculate loss function and score using Python
- recognize the critical causes of optimization problems and the essential alternatives to optimization
- recall the prominent optimizer algorithms along with their properties that can be applied for optimization
- demonstrate how to perform comparative optimizer analysis using Keras
- recognize the relevance of learning rates in optimization and list the various approaches of improving learning rates
- demonstrate the approach of finding learning rate using RMSProp optimizer
- recall the different types of loss functions, list the prominent cause of optimization problems, and calculate loss function using Python

<table>
<tr>
<td> 

Improving Neural Networks: Data Scaling & Regularization Neural Networks</td>
<td> 

ConvNets: Introduction to Convolutional Neural Networks</td>
</tr>
<tr>
<td>Objectives

▪ discover the key concepts covered in this course
▪ describe the approach of creating deep learning network models along with the steps involved in optimizing the networks
▪ implement the learning rate adaptation schedule in Keras using SGD and specifying learning rate, epoch and decay
▪ describe the concept of scaling data and list the prominent data scaling methods
▪ describe the concept of batch normalization and internal covariate shift
▪ demonstrate how to implement batch normalization using Python and TensorFlow
▪ implement L1 regularization to manage overfitting problems
▪ implement L2 regularization to manage overfitting problems
▪ demonstrate how to implement gradient descent using Python
▪ recall the prominent data scaling methods, implement L1 regularization and gradient descent using Python</td>
<td>Objectives

▪ discover the key concepts covered in this course
▪ define the concept of convolution neural networks and recognize the prominent uses cases of convolutional neural networks
▪ describe the working mechanism of convolutional neural networks
▪ recognize the different types of convolutional neural networks that we can implement
▪ describe the problems associated with computer vision along with the prominent techniques to manage them
▪ identify the role of neural networks and convolutional neural networks in implementing and addressing image recognition and classification problems
▪ recognize the prominent layers and parameters of convolutional neural networks for image classification
▪ describe convolutional layer from mathematical perspective and recognize the mathematical elements that enter into the convolution operations
▪ build a convolutional neural network for image classification using Python
▪ recall the prominent use cases of convolutional neural networks, list the different types of convolutional neural networks, and build a simple convolutional neural network for image classification</td>
</tr>
</table>

ConvNets: Working with Convolutional Neural Networks

Objectives

- discover the key concepts covered in this course
- recall the architecture of neural networks along with the essential elements used for computations with focus on softmax classifier
- work with ConvNetJS as a Javascript library and train deep learning models
- define the concept of the edge detection method and list the common algorithms that are used for edge detection
- recognize the series of convolution and pooling operations to detect features
- recognize the involvement of maths in convolutional neural networks and recall the essential rules that are applied on filters and channel detection
- illustrate the principle of convolutional layer, activation function, pooling layer and fully-connected layer
- recognize the need for activation layer in convolutional neural networks and compare the prominent activation functions for deep neural networks
- recall the different approaches to improve convolution neural networks and machine learning systems
- list the common algorithms that are used for edge detection, recall the essential rules that are applied on filters and channel detection and specify some of the critical approaches that we can adopt to improve convolutional neural networks

Convolutional Neural Networks: Fundamentals

Objectives

- illustrate the concept of visual signal perception using a biological example
- describe convolutional neural network, its architecture, and its layers
- describe the driving principles of convolutional neural network
- describe the combined approach of implementing convolutional layer and sparse interaction
- describe shared parameters and spatial in a convolutional neural network (CNN)
- describe convolutional padding and strides in a convolutional neural network (CNN)
- recognize the relevance and importance of pooling layers in convolutional neural networks (CNNs)
- use ReLU on convolutional neural networks (CNNs)
- define semantic segmentation and its implementation using Texton Forest and random-based classifier
- describe gradient descent and list its prominent variants
- list CNN layers, implementation approaches, layers, and variants of gradient descent
-

 Convolutional Neural Networks: Implementing & Training	 Convo Nets for Visual Recognition: Filters and Feature Mapping in CNN
Objectives ▪ implement image recognition classifier using convolutional neural networks, Keras, and TensorFlow ▪ describe ResNet layers and blocks ▪ list the essential PyTorch ecosystem tools that can be used to develop and implement neural networks ▪ install and configure PyTorch ▪ implement convolutional neural networks (CNNs) using PyTorch ▪ use PyTorch to train convolutional neural networks (CNNs) to categorize garments ▪ install and configure PyTorch and implement convolutional neural networks (CNNs) using PyTorch	Objectives ▪ discover the key concepts covered in this course ▪ recognize the capability and features of convolutional networks that makes it a recommended choice for visual recognition implementation ▪ illustrate the architecture and the various layers of convolutional networks ▪ define the concept and types of filters in convolutional networks along with their usage scenarios to depict the impact of filters on feature sets during the training process ▪ demonstrate the approach of using Keras to visualize inputs that maximize the activation of filters in different layers of convolutional networks ▪ define the concept of feature map in convolutional networks and illustrate the approach of visualizing feature maps ▪ plot the feature map of the first convo layer for given images and visualize the feature map output from every block in the VGG model ▪ identify the critical parameters that we need to tune to optimize convolutional networks ▪ recall the essential hyperparameters that are applied on convolutional networks for optimization and model refinement ▪ work with hyperparameters using Keras and TensorFlow to derive optimized convolutional network models ▪ recognize the role of pooling layer in convolutional networks along with the various operations and functions that we can apply on the layer ▪ demonstrate how to implement convo and pooling layer in Python ▪ recall the various layers of convolutional networks, plot the feature map of the first convo layer for a given image and visualize the Feature map output from every block in the VGG model

<table>
<tr>
<td> 

Convo Nets for Visual Recognition: Computer Vision & CNN Architectures</td>
<td> 

Fundamentals of Sequence Model: Artificial Neural Network & Sequence Modeling</td>
</tr>
<tr>
<td>Objectives

▪ discover the key concepts covered in this course
▪ define and classify activation functions and provide a comparative analysis with the pros and cons of the different types of activation functions
▪ recognize the limitations of Sigmoid and Tanh and describe how they can be resolved using ReLU along with the significant benefits afforded by ReLU when applied in convolutional networks
▪ implement rectified linear activation function in convolutional networks using Python
▪ list the core tasks that are used in the implementation of computer vision
▪ develop convolutional neural network models from the scratch for object photo classification using Python and Keras
▪ describe the concept of fully-connected layer and illustrate its role in convolutional networks
▪ describe the convolutional neural network training process workflow and the essential elements that we need to specify during the training process
▪ list and compare the various architectures of convolutional neural networks
▪ recall the benefits of applying ReLU in convolutional neural networks, list the prominent architectures of convolutional neural networks and implement ReLU function in convolutional networks using Python</td>
<td>Objectives

▪ describe artificial neural networks (ANNs) and their features and characteristics
▪ list artificial neural network components used to build a model
▪ list prominent tools and frameworks used implement sequence models and artificial neural networks
▪ describe sequence modeling as it pertains to language models
▪ describe recurrent neural networks and their capabilities and components
▪ specify RNN types and their implementation features
▪ build a recurrent neural network using PyTorch and Google Colab
▪ recall ANN characteristics, modeling tools, and architectures and applications of sequence models</td>
</tr>
</table>

Fundamentals of Sequence Model: Language Model & Modeling Algorithms

Objectives

- describe language models and how to implement natural language processing (NLP) and its components
- generate sequences for natural language processing (NLP)
- describe vanishing gradient problem implementation approaches
- list features and characteristics of gated recurrent units (GRUs)
- recognize the problems and drawbacks of implementing short-term memory and LSTM as modeling solutions
- list characteristics of language modeling, vanishing gradient problems, and LSTM

Build & Train RNNs: Neural Network Components

Objectives

- describe artificial neural network and its components
- identify the topology of the networks that implements feedforward, recurrent and linked networks
- list activation mechanisms used in the implementation of neural networks
- specify the prominent learning samples that can be applied in neural networks
- compare Supervised, Unsupervised, and Reinforcement learning samples
- describe training samples and the approaches for building them
- identify training sets and recognize patterns
- recognize the need for gradient optimization in neural networks
- list neural network components, activation functions, learning samples, and gradient descent optimization algorithms

Build & Train RNNs: Implementing Recurrent Neural Networks

Objectives

- identify the essential features and processes of implementing perception and backpropagation
- compare single and multilayer perception and describe the need for layer management
- describe the steps involved in building recurrent neural network models
- implement recurrent neural network using Python and TensorFlow
- implement long short-term memory using TensorFlow
- recognize the capabilities provided by Caffe2 for implementing neural networks
- implement recurrent neural network using Caffe2
- build deep learning language models using Keras
- implement RNN using TensorFlow and Caffe2 and build deep learning language models using Keras

ML Algorithms: Multivariate Calculation & Algorithms

Objectives

- recognize the role of multivariate calculus in machine learning
- describe functions in calculus
- define the concepts of gradient and derivative and describe their applications on the functions of variables
- list the capabilities of the product and chain rules
- define partial differentiation and its application in vector calculus and differential geometry
- recognize the importance of linear algebra in machine learning
- describe optimization techniques when using Gradient and Jacobian matrix
- define Taylor's theorem and specify the conditions for local minima
- list various multivariate operations that can be used in multivariate calculus, compare the differences between a gradient and derivative, recall examples of partial differential equation, and specify the domains where linear algebra is implemented

ML Algorithms: Multivariate Calculation & Algorithms

Objectives

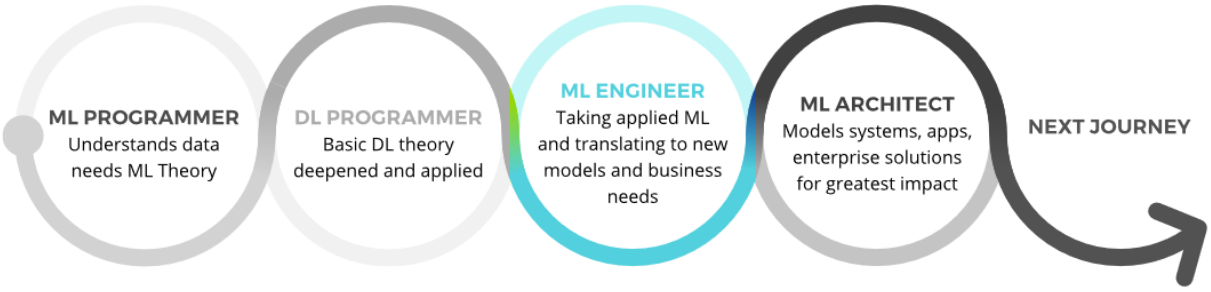
- recognize the role of multivariate calculus in machine learning
- describe functions in calculus
- define the concepts of gradient and derivative and describe their applications on the functions of variables
- list the capabilities of the product and chain rules
- define partial differentiation and its application in vector calculus and differential geometry
- recognize the importance of linear algebra in machine learning
- describe optimization techniques when using Gradient and Jacobian matrix
- define Taylor's theorem and specify the conditions for local minima
- list various multivariate operations that can be used in multivariate calculus, compare the differences between a gradient and derivative, recall examples of partial differential equation, and specify the domains where linear algebra is implemented

DL Programming with Python

Final Exam: DL Programmer

DATA JOURNEY

MACHINE LEARNING PROGRAMMER TO MACHINE LEARNING ARCHITECT



Track 3: ML Engineer

 Predictive Modeling: Predictive Analytics & Exploratory Data Analysis	 Predictive Modeling: Implementing Predictive Models Using Visualizations
Objectives ▪ define the predictive analytics and describe its process flow ▪ describe analytical base table and how it can be used to build and score analytical models ▪ identify the business problems that can be resolved using predictive modeling ▪ build predictive models using the Python framework ▪ list essential features of exploratory data analysis ▪ describe univariate, bivariate, and multivariate data and analytical approaches that can be implemented with them ▪ specify methods that can be used to manage missing values and outliers in datasets ▪ list applications of predictive analytics, describe analytical base tables, list predictive models, and specify variable selection methods	Objectives ▪ list the benefits of feature selection and the general classes of feature selection algorithms ▪ recall the different types of predictive models that can be implemented and features ▪ implement scatter plots and describe the capability of scatter plots in facilitating predictions ▪ define Pearson's correlation measures and specify the possible ranges for Pearson's correlation ▪ recognize the anatomy of a boxplot ▪ create and interpret boxplots using Python ▪ implement crosstabs to visualize categorical variables ▪ describe statistical concepts that are used for predictive modeling ▪ demonstrate the tree-based methods that can be used to implement regression and classification ▪ describe the best practices for implementing predictive modeling ▪ implement boxplots, scatter plots, and crosstabs using Python

 Applying Predictive Analytics	 Planning AI Implementation
Objectives ▪ describe predictive analytics is and where it is applicable ▪ recognize the predictive modeling process, including how to explore and understand data, prepare for and model data, and evaluate and deploy the model ▪ identify methods for random sampling and use hypothesis testing, Chi-square tests, and correlation ▪ recognize common model categories and analytical techniques ▪ use Decision Trees and Support Vector Machines for predictive analytics ▪ use Survival Analysis to predict and analyze customer churn ▪ use Market Basket Analysis to perform predictive analysis ▪ apply data clustering models to perform predictive analysis ▪ apply random forests for predictive analytics ▪ use Naive Bayes classifiers for predictive analytics ▪ use Logistic Regression for predictive analytics ▪ describe best practices in predictive modeling ▪ apply models to perform predictive analytics ▪	Objectives ▪ describe how adopting an AI strategy requires proper expectations and buy-in ▪ specify the various organizational challenges surrounding the adoption of AI ▪ describe personnel training and how an AI implementation requires training ▪ describe the role that data and algorithms play in an organizational AI implementation ▪ specify why AI is commonly misinterpreted as substitution for personnel and how it should be considered as a complement to personnel ▪ describe the ways that an organization can plan and develop AI capability ▪ describe the challenges facing management when developing an AI solution and how it can impact personnel ▪ specify the pitfalls of AI and why they should be avoided ▪ describe the common elements of an organizational AI strategy ▪ specify the various issues surrounding data, data quality, and the concepts of training, overfitting, and bias ▪ describe the elements organizations need to understand in order to assess AI needs and tools ▪ discuss the various aspects of AI an organization needs to address to plan for AI

<table>
<tr>
<td> 

Automation Design & Robotics</td>
<td> 

ML/DL in the Enterprise: Machine Learning Modeling, Development, & Deployment</td>
</tr>
<tr>
<td>Objectives

▪ describe automation and how it is implemented
▪ describe the tasks and processes best suited for automation
▪ describe the Display Status automation design principle
▪ describe the Human-Computer Collaboration automation design principle
▪ describe the Human Intervention automation design principle
▪ describe automated testing in software design and development
▪ describe task runners in software design and development
▪ describe DevOps and automated deployment in software design, development, and deployment
▪ describe software design patterns in robotics
▪ describe the robotics process automation practice
▪ recognize how modern robotics and AI designs are applied
▪ recognize automation and robotics design application</td>
<td>Objectives

▪ discover the key concepts covered in this course
▪ list critical challenges that may be encountered when implementing machine learning
▪ recognize the essential stages of machine learning processes that need to be adopted by enterprises
▪ describe the development lifecycle exclusively used to build productive machine learning models
▪ specify the essential phases of machine learning workflows and the functional flow of each phase
▪ recall the critical processes that are involved in training machine learning models
▪ list the various on-premise and cloud platforms that can be used to develop and deploy machine learning projects
▪ describe the approaches that can be adopted to model and process data for productive machine learning deployments
▪ set up a machine learning development and deployment environment using H2O clusters
▪ recall the various data stores and data management frameworks that can be used as a data layer for machine learning environments
▪ specify the processes involved in implementing machine learning pipelines and using visualizations to generate insights
▪ set up and work with Git to facilitate team-driven development and coordination across the enterprise
▪ specify processes involved in training machine learning models, recall the prominent cloud platforms used to build and deploy machine learning projects, and set up machine learning deployment environment on AWS</td>
</tr>
</table>

<table>
<tr>
<td> 

ML/DL in the Enterprise: Machine Learning Infrastructure</td>
<td> 

Enterprise Services: Enterprise Machine Learning with AWS</td>
</tr>
<tr>
<td>Objectives

▪ discover the key concepts covered in this course
▪ list critical challenges that may be encountered when implementing machine learning
▪ recognize the essential stages of machine learning processes that need to be adopted by enterprises
▪ describe the development lifecycle exclusively used to build productive machine learning models
▪ specify the essential phases of machine learning workflows and the functional flow of each phase
▪ recall the critical processes that are involved in training machine learning models
▪ list the various on-premise and cloud platforms that can be used to develop and deploy machine learning projects
▪ describe the approaches that can be adopted to model and process data for productive machine learning deployments
▪ set up a machine learning development and deployment environment using H2O clusters
▪ recall the various data stores and data management frameworks that can be used as a data layer for machine learning environments
▪ specify the processes involved in implementing machine learning pipelines and using visualizations to generate insights
▪ set up and work with Git to facilitate team-driven development and coordination across the enterprise
▪ specify processes involved in training machine learning models, recall the prominent cloud platforms used to build and deploy machine learning projects, and set up machine learning deployment environment on AWS</td>
<td>Objectives

▪ discover the key concepts covered in this course
▪ recognize cloud features that provide significant operational benefits for implementing machine learning
▪ describe the machine learning workflow and differentiate between machine learning model development and traditional enterprise software development
▪ recall the machine learning tools, services, and capabilities provided by AWS
▪ compare machine learning implementation scenarios and solutions in AWS, Microsoft Azure, and Google Cloud to be able to identify the best fit for any given scenario
▪ describe the machine learning objects and the mechanisms of generating and interpreting predictions available with AWS
▪ use the Amazon Machine Learning console to create data sources and build machine learning models, and use the models to generate predictions
▪ describe the architecture of Amazon SageMaker as well as the internal AWS components used in Amazon SageMaker with focus on algorithm, training, and hosting services
▪ use the Amazon SageMaker to create, train, and deploy simple machine learning models
▪ describe the features of Lex, Polly, and Transcribe and their roles in machine learning implementation
▪ recognize the features of Amazon SageMaker Neo that enable machine learning models to train once and run anywhere
▪ use Augmented Manifest to train object detection machine learning model with Amazon SageMaker
▪ describe the automatic model tuning capabilities of Amazon SageMaker that are applied for hyperparameter tuning functionality
▪ use Amazon SageMaker for hyperparameter tuning and use the pre-built TensorFlow container and MNIST dataset to tune models
▪ summarize the key concepts covered in this course</td>
</tr>
</table>

Enterprise Services: Machine Learning Implementation on Microsoft Azure

Objectives

- discover the key concepts covered in this course
- describe Azure machine learning tools, services, and capabilities
- compare the capabilities of Azure Machine Learning Studio and Azure Machine Learning Service
- create Azure Machine Learning Service workspaces and configure development environments for Azure Machine Learning
- build and manage machine learning pipelines with Azure Machine Learning Service
- launch the Microsoft Azure Machine Learning Studio and work with datasets, train models, and projects
- use the Azure Machine Learning Service visual interface to develop and deploy predictive analytic solutions
- access, transform, and join data using Azure Open Datasets and train automated machine learning regression models to calculate model accuracy
- describe the capabilities of MLOps with focus on managing, deploying, and monitoring models using Azure Machine Learning Service to improve the quality and consistency of machine learning solutions
- work with Azure Machine Learning R Notebooks to fit models and publish models as web services
- build predictive pipelines, incorporating Azure Data Lake and Azure Machine Learning
- enable CI/CD for machine learning projects with Azure Pipelines
- use the ML extension of Visual Studio from Microsoft DevLabs to track code from Azure Repos or GitHub, trigger release pipelines, and automate machine learning deployments using Azure Pipelines
- summarize the key concepts covered in this course

Enterprise Services: Machine Learning Implementation on Google Cloud Platform

Objectives

- discover the key concepts covered in this course
- describe GCP machine learning tools, services, and capabilities
- describe the Google Cloud Platform machine learning implementation approach and the different stages in Google Cloud Platform machine learning workflow
- train models using the built-in linear learner algorithm and submit jobs with GCloud and Console
- recall the essential features of BigQuery along with the capabilities of BigQuery ML
- create and evaluate binary logistic regression models using BigQuery ML statements
- recognize the challenges associated with modern machine learning workflows and how you can leverage the serverless approach to eliminate those challenges
- set up and work with Cloud Datalab
- recognize Google AI Platform components and features that can be used to build machine learning workflows and train machine learning models at scale
- recall Google Cloud AutoML features and how it can be used to train, evaluate, and deploy machine learning models
- use AutoML Tables to work with datasets needed to train and use machine learning models
- work with AutoML Tables to train machine learning models using imported datasets and use the trained models for predictions
- work with Google Cloud AutoML Natural Language to create custom machine learning models for content category classification
- summarize the key concepts covered in this course

<table>
<tr>
<td> 
</td>
<td>Enterprise Architecture: Architectural Principles & Patterns</td>
<td> 
</td>
<td>Enterprise Architecture: Design Architecture for Machine Learning Applications</td>
</tr>
<tr>
<td colspan="2">Objectives

▪ discover the key concepts covered in this course
▪ describe software architecture and the benefits it provides
▪ list the principles that should be followed when designing architectures for applications
▪ describe the 4+1 view model and the associated views
▪ recognize the software architectures that are used to manage applications from the perspective of the end user and technologies
▪ list principles that can be used when developing enterprise architecture along with the factors that influence the principles
▪ recall the prominent architectural principles that can be applied on business, data, and technology
▪ recognize the fundamental principles guiding the adoption of SOA and the usage of the SOA maturity model
▪ recall the benefits of adopting serverless architecture over traditional compute intensive architecture
▪ describe Backend-as-a-Service and the architectural components that are used to derive it and integrate it with client-focused applications
▪ describe evolutionary architectures and their features
▪ recognize the benefits of documenting architectures and documentation elements that help in depicting architectural components
▪ describe the structure of a software project team and the impact of collaboration on enterprise software architecture
▪ describe the concept and characteristics of consumer-driven contracts, which are used to manage the challenges in the community of service providers and consumers
▪ specify the dimensions of the architecture that should be coupled to provide maximum benefit with minimal overhead and cost
▪ recognize the activities and tasks that software architects perform
▪ specify the architectural patterns, styles, and solution patterns that can be adopted to eliminate common problems within given contexts
▪ summarize the key concepts covered in this course</td>
<td colspan="2">Objectives

▪ discover the key concepts covered in this course
▪ describe the basic architecture required to execute machine learning implementations in the enterprise
▪ describe software architectures and their associated features that can be used to model machine learning applications in production
▪ set up an enterprise architecture to implement a robust memory model
▪ describe machine learning reference architecture and the associated building blocks of the reference architecture
▪ describe approaches for building evolvable architectures and migrating architectures
▪ recognize the pitfalls of evolutionary architecture and the antipatterns of technical architecture and incremental change
▪ demonstrate how to set up complete machine learning solutions
▪ describe the Fitness function and its associated categories
▪ recognize architectural planning guidelines for machine learning projects, with focus on model refinement, testing, and evaluating production readiness
▪ summarize the key concepts covered in this course
▪</td>
</tr>
</table>

Architecting Balance: Designing Hybrid Cloud Solutions

Objectives

- discover the key concepts covered in this course
- recognize the essential features afforded by cloud and cloud deployment models
- compare the characteristics of on-premise and cloud deployment from the perspective of cost, hardware, software, mobility, and reliability
- identify the critical factors that can influence decisions of implementing either a cloud or on-premise architecture
- recall the prominent components of hybrid cloud that are required to build hybrid environment
- use hybrid cloud assessment to identify appropriate scenarios for adopting a hybrid cloud architecture
- list the essential factors to consider when deriving a hybrid cloud implementation strategy and architecture
- recognize the critical advantages and benefits that businesses can gain by adopting a hybrid cloud architecture
- specify the challenges and the key considerations that can help mitigate the challenges of successfully implementing a hybrid cloud architecture
- identify the key considerations that can help derive strategies to determine whether to move and deploy applications to the cloud or retain them on on-premise environment
- demonstrate the steps involved in setting up deployment architectures for deploying three tier applications
- recall the essential benefits and advantages afforded by hybrid cloud, and specify the difference between on-premise and hybrid cloud deployment models

Architecting Balance: Hybrid Cloud Implementation with AWS & Azure

Objectives

- discover the key concepts covered in this course
- recognize the critical use cases of implementing hybrid cloud using AWS
- list the prominent AWS services that can be used to implement hybrid cloud solutions
- describe the cloudbursting application hosting model from the perspective of AWS
- recall the essential AWS services that provide integrated resource and deployment management capabilities for hybrid cloud solutions
- implement a hybrid cloud environment that integrates on-premise Hadoop clusters with data lakes on AWS
- recall the recommended principles and best practices for AWS hybrid cloud implementation from the perspective of provisioning, operations, management, and monitoring
- list the essential components provided by Azure that can be used to implement hybrid cloud solutions by integrating Azure with on-premise infrastructure
- recognize the essential Azure hybrid cloud capabilities from the perspective of networking, application, data, identity, and management
- illustrate the features and architecture of Azure Stack, along with its role in a hybrid cloud solution implementation
- recall the prominent tooling and DevOps services that can be used with Azure and Azure Stack to implement hybrid cloud
- demonstrate how to use Azure Stack to implement hybrid cloud and deploy applications
- recall the prominent AWS services that can be used to implement hybrid cloud, specify the essential components provided by Azure that can be used to implement hybrid cloud and implement hybrid cloud using Azure Stack

Refactoring ML/DL Algorithms: Techniques & Principles

Objectives

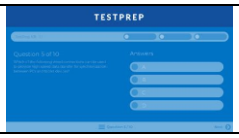
- discover the key concepts covered in this course
- define the concept of refactoring and the primary coding issues that can be resolved with code refactoring
- recognize the causes of technical debts, which contribute to coding challenges
- list code refactoring techniques and types of processes that can be used to eliminate code deficiencies
- demonstrate the refactoring capabilities provided by Pycharm to refactor Python code
- demonstrate static code analysis for Python using Prospector and refactor code to ensure backward compatibility
- illustrate the role of design patterns in code refactoring
- recall the essential principles and challenges associated with code refactoring
- recognize critical code elements and corresponding examples to illustrate the principles of good coding
- recognize techniques that can be used to refactor Python code and derive the best coding outcomes
- demonstrate the steps involved in optimizing Python code
- demonstrate how to use Rope to refactor Python code
- recall anti-patterns that can potentially complicate code and code refactoring
- recall code refactoring techniques, specify the principles and challenges associated with code refactoring, and use Rope to refactor Python code

Refactoring ML/DL Algorithms: Refactor Machine Learning Algorithms

Objectives

- discover the key concepts covered in this course
- describe approaches for selecting an appropriate machine learning implementation
- specify the steps involved in designing machine learning algorithms
- describe the impact of refactoring machine learning code
- recognize the principles of designing machine learning algorithms
- compare prominent machine learning algorithms and select the appropriate algorithm for diversified problem spaces
- demonstrate how to refactor existing machine learning code that is written in Python
- identify the essential approaches of managing technical debts in machine learning implementations
- use SonarQube to build code coverage for machine learning code that is written in python
- describe the approach of automatic clone recommendation for refactoring based on the present and the past
- recall the principles involved in designing machine learning algorithms and refactor machine learning code written in Python

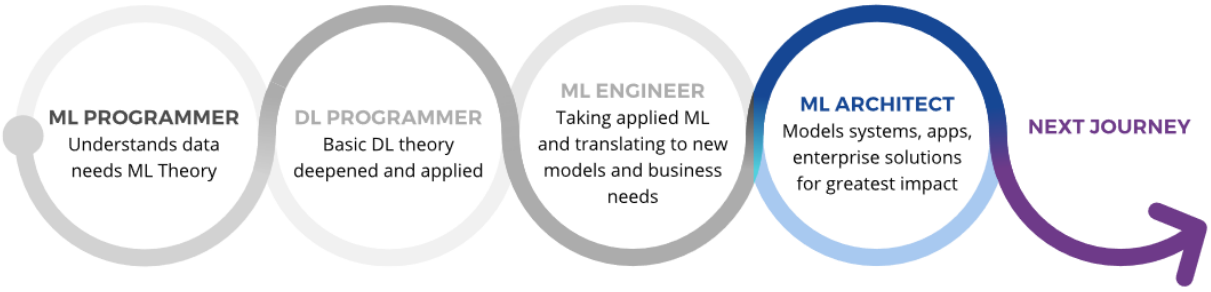
Practice Lab: ML Engineer



Final Exam: ML Engineer

DATA JOURNEY

MACHINE LEARNING PROGRAMMER TO MACHINE LEARNING ARCHITECT



Track 4: ML Architect (duration: 12h 27m 38s)

 Applied Predictive Modeling	 Implementing Deep Learning: Practical Deep Learning
Objectives ▪ recognize characteristics of predictive modeling ▪ use Python and related data analysis libraries to perform exploratory data analysis ▪ recognize key features of Linear and Logistic regressions ▪ apply a linear regression with Python ▪ apply a logistic regression with Python ▪ recognize key features of hierarchical clustering and K-Means clustering ▪ apply hierarchical clustering with Python ▪ apply K-Means clustering with Python ▪ recognize key features of Decision Trees and Random Forests ▪ apply a Decision Tree with Python ▪ apply a Random Forest with Python ▪ apply linear regression, logistic regression, hierarchical clustering, Decision Trees, and Random Forests with Python	Objectives ▪ discover the key concepts covered in this course ▪ define the concept of deep learning and compare the differences between machine learning and deep learning ▪ describe the various phases of ML/DL workflows involved in building deep learning networks ▪ recall the essential components of building and applying deep learning networks ▪ list the prominent frameworks that can be used to simplify building ML/DL applications ▪ use the Caffe2 framework to build recurrent convolution neural networks ▪ write PyTorch code to generate images using autoencoders ▪ implement deep neural networks using Python and Keras ▪ compare the prominent platforms and frameworks that can be used to simplify deep learning implementations ▪ identify and select the best fit frameworks for prominent ML/DL use cases ▪ recognize the challenges and strategies associated with debugging deep learning networks and algorithms ▪ identify steps of machine learning workflow, deep learning frameworks, and strategies for debugging deep learning networks ▪

Implementing Deep Learning: Optimized Deep Learning Applications

Objectives

- discover the key concepts covered in this course
- define the concept of computational graphs and recognize the essential computational graph operations that are used in implementing deep learning
- list the prominent processors with specialized purpose and architectures that are used in implementing deep learning
- recall the prominent interfaces for programming GPUs with focus on CUDA and OpenCL
- define the concept of TensorFlow Extended and list the essential TFX pipeline components that can be used to implement machine learning pipelines
- setup the TensorFlow Extended environment to build deep learning pipelines
- demonstrate how to use the ExampleGen and StatisticsGen TFX pipeline components to build pipelines
- work with TensorFlow Model Analysis to investigate and visualize the characteristics of datasets and the performances of models
- recognize the practical features and elements that should be considered when building deep learning models, with focus on baseline model and optimization
- recall the essential hyperparameters of deep learning algorithms that can be tuned to optimize deep learning models
- identify components of a computation graph, common GPU frameworks, and tasks that can improve performance with data

Applied Deep Learning: Unsupervised Data

Objectives

- discover the key concepts covered in this course
- recall the concept of deep learning and the approach of using deep learning-based frameworks to model NLP tasks and audio data analysis
- describe the role of recurrent neural network and the various architectures of recurrent neural network that can be used in modeling natural language processing
- recognize the challenges associated with unsupervised learning and the approach of using deep learning from the perspective of common unsupervised feature learning
- describe the prominent statistical classification models and compare generative classifiers with discriminative classifiers
- recall the different types of generative models, with focus on generative adversarial network, variational autoencoders, and flow-based generative model
- demonstrate the steps involved in setting up and working with PixelCNN
- describe the characteristics of the different classes of artificial neural networks and the difference between MLP, CNN, and RNN
- recognize the essential capabilities and variants of ResNet that can be used for computer vision and deep learning
- describe encoders in neural networks and compare the capabilities of standard autoencoders and variational autoencoders
- recall the prominent architectures of recurrent neural network that can be used in modeling natural language processing, list the variants of ResNet that can be used for computer vision and deep learning, and set up PixelCNN

<table>
<tr>
<td> 

Applied Deep Learning: Generative Adversarial Networks and Q-Learning</td>
<td> 

Advanced Reinforcement Learning: Principles</td>
</tr>
<tr>
<td>Objectives

▪ discover the key concepts covered in this course
▪ use deep convolutional autoencoder with Keras and Python
▪ implement generative adversarial network and the role of Generator and Discriminator
▪ implement generative adversarial network Discriminator and Generator using Python and Keras and build Discriminator for training model
▪ recognize the challenges of working with generative adversarial network models
▪ describe the concept of deep reinforcement learning and its application in the areas of robotics, finance, and healthcare
▪ compare deep reinforcement learning with deep learning, and describe the challenges associated with their implementations
▪ describe the basic concepts of reinforcement learning, as well as the concept of deep Q-learning and its implementation
▪ implement deep Q-learning in Python using Keras and OpenAI Gym
▪ recall the variations of generative adversarial network, implement generative adversarial network Discriminator and Generator using Python, and implement deep Q-learning in Python using Keras and OpenAI Gym</td>
<td>Objectives

▪ discover the key concepts covered in this course
▪ define reinforcement learning and the important terms that are used to formulate reinforcement learning problems
▪ differentiate between the implementations of reinforcement and machine learning using supervised and unsupervised learning
▪ describe the capabilities of reinforcement learning, illustrating its uses cases and example implementations
▪ recognize reinforcement learning terms that are used in building reinforcement learning workflows
▪ describe approaches of implementing reinforcement learning
▪ describe reinforcement learning algorithms and their features
▪ describe Markov Decision Process, its variants, and the steps involved the algorithm
▪ describe Markov Reward Process, with focus on value functions for implementing Markov reward process
▪ recognize the capabilities of the Markov Decision Process toolbox and the algorithms that are implemented within it
▪ recall the reinforcement learning terms, describe reinforcement learning implementation approaches, and list the Markov Decision Process algorithms</td>
</tr>
</table>

<table>
<tr>
<td> 

Advanced Reinforcement Learning: Implementation</td>
<td> 

ML/DL Best Practices: Machine Learning Workflow Best Practices</td>
</tr>
<tr>
<td>Objectives

▪ discover the key concepts covered in this course
▪ install the Markov Decision Policy toolbox and implement the Discounted Markov Decision Process using the policy iteration algorithm
▪ recognize the role of reward and discount factors in reinforcement learning
▪ describe the multi-armed bandit problem and different approaches of solving this problem
▪ describe dynamic programming, policy evaluation, policy iteration, value iteration, and characteristics of Bellman equation
▪ list reinforcement learning agent components and reinforcement agent applications
▪ work with reinforcement learning agents using Keras and OpenAI Gym
▪ describe reinforcement learning algorithms and the reinforcement learning taxonomy defined by OpenAI
▪ implement deep reinforcement learning or deep Q-learning using Keras and OpenAI Gym
▪ recognize how to train deep neural networks using reinforcement learning for time series forecasting
▪ recall approaches for resolving the multi-armed bandit problem, list reinforcement learning agent components, and implement deep Q-learning using Keras and OpenAI Gym</td>
<td>Objectives

▪ discover the key concepts covered in this course
▪ list the various phases of machine learning workflow that can be used to achieve key milestones of machine learning projects
▪ recall the data workflows that are used to develop machine learning models
▪ identify the differences between machine learning and deep learning and illustrate the phases of deep learning workflow
▪ list the best practices that should be adopted to build robust machine learning systems, with focus on the evaluation approach
▪ recall the challenges of debugging machine learning and deep learning projects and the factors that need to be considered while debugging
▪ describe the approach of debugging trained machine learning models using flippoints
▪ recognize the benefits of implementing machine learning checklists and the process of building checklists that can be used to work through applied machine learning problems
▪ describe checklists for debugging neural networks, the steps involved in identifying and fixing issues associated with training, and generalizing and optimizing machine learning models
▪ recall the checklists for implementing end-to-end machine learning and deep learning workflows that should be adopted to build optimized machine learning and deep learning models
▪ describe the high-level deep learning strategies and the common design choices for implementing deep learning projects
▪ summarize the key concepts covered in this course</td>
</tr>
</table>

<table>
<tr>
<td> 

ML/DL Best Practices: Building Pipelines with Applied Rules</td>
<td> 

Research Topics in ML & DL</td>
</tr>
<tr>
<td>Objectives

▪ discover the key concepts covered in this course
▪ list deep learning model troubleshooting steps and recommended data and model checklists for building robust deep learning solutions
▪ recognize machine learning technical challenges and the best practices for dealing with the identified challenges
▪ use case studies to analyze the impacts of adopting best practices for deep learning
▪ identify the challenges and patterns associated with deploying deep learning solutions in the enterprise
▪ describe approaches for deploying deep learning solutions in the enterprise using case study scenarios
▪ describe approaches for architecting and building machine learning pipelines to implement scalable machine learning systems
▪ specify the rules that should be applied while building machine learning pipelines into applications
▪ identify the rules that should be applied when using feature engineering to pull the right features into applications
▪ specify the causes of training-serving skew and the rules that should be considered to manage training-serving skew
▪ define the rules for managing slowed growth, optimization refinement, and complex models in machine learning application management
▪ describe checklists for machine learning projects that are to be prepared and adopted by project managers
▪ summarize the key concepts covered in this course</td>
<td>Objectives

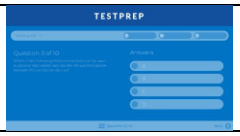
▪ understand the efforts being undertaken to reduce overfitting using the dropout technique
▪ understand leading edge multi-label learning algorithms
▪ understand the proposed learning framework for deep residual learning that improves training of networks that are significantly deeper than traditional neural networks
▪ understand how initializing a network with transferred features may boost generalization performance
▪ understand how convolutional neural networks may be utilized as a powerful class of models for image recognition
▪ understand the dataset that advances state-of-the-art object recognition by considering the context within the question of scene understanding
▪ understand the proposed framework for estimating generative models via an adversarial process that successfully estimates the probability that a sample came from training data rather than a generative model
▪ understand how optimal nearest neighbor algorithms perform compared to traditional nearest neighbor algorithms
▪ understand how an ensemble of regression trees may successfully estimate facial landmark positions while delivering real-time performance and high quality predictions
▪ understand how a proposed new scene-centric database is successfully used for learning deep features for scene recognition
▪ recognize how ELM tends to produce better scalability, generalization performance, and faster learning than traditional support vector machine
▪ understand the trending research topics in ML and DL</td>
</tr>
</table>

Deep Learning with Keras

Objectives

- describe what neural networks are and their main components
- describe Keras and its guiding principles
- configure CNTK as your Keras backend
- install and configure Keras
- identify and work with both types of models available in Keras
- recognize features of commonly used Keras layers and when to use them
- use Keras to make regression classifications
- use Keras to make image classifications
- use Keras metrics to judge the performance of your model
- use Jupyter Notebooks with Keras
- download and load a dataset from MNIST or CIFAR-10
- explore your dataset in Keras
- prepare your data in Keras by defining your input and target tensors
- compile the model in Keras
- train and test your neural network
- evaluate and score the performance of your neural network in Keras
- make predictions using your dataset in Keras
- use a neural network to make predictions

LAB: ML Architect



Final Exam - ML Architect

Business & Leadership Skills

Business & Leadership for ML Architects

Optional

COURSE
Getting to the Root of a Problem
298

COURSE
Confronting Your Assumptions
182

COURSE
Investigating Arguments
93

COURSE
Managing a Project to Minimize Risk and Maximiz...
92

COURSE
Embracing an Agile Culture for Business Growth
86

COURSE
Leading through Positive Influence
192

COURSE
Leading a Cross-functional Team
56

COURSE
Cultivating Relationships with Your Peers
71

COURSE
The Building Blocks of Building Trust
135

COURSE
Trust Building through Effective Communication
304

COURSE
Defining Alternative Solutions to a Problem
103

COURSE
Six Sigma Performance Metrics
28

COURSE
Thinking Strategically as a Manager
220

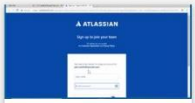
COURSE
Cultivating Cross-functional Team Collaboration
25

COURSE
Unleashing Personal and Team Creativity
192

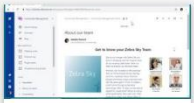
Productivity Tools for ML Architects

Productivity Tools for ML Architects

Optional

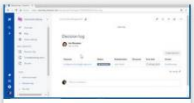


COURSE
Signing in & Navigating within Spaces
11



COURSE
Setting Up & Managing Spaces
9

COURSE
Working with Spaces
7



COURSE
Working with Team Members
27

COURSE
Configuring Spaces
7

COURSE
Signing in & Setting Up
13

COURSE
Using Channels
6

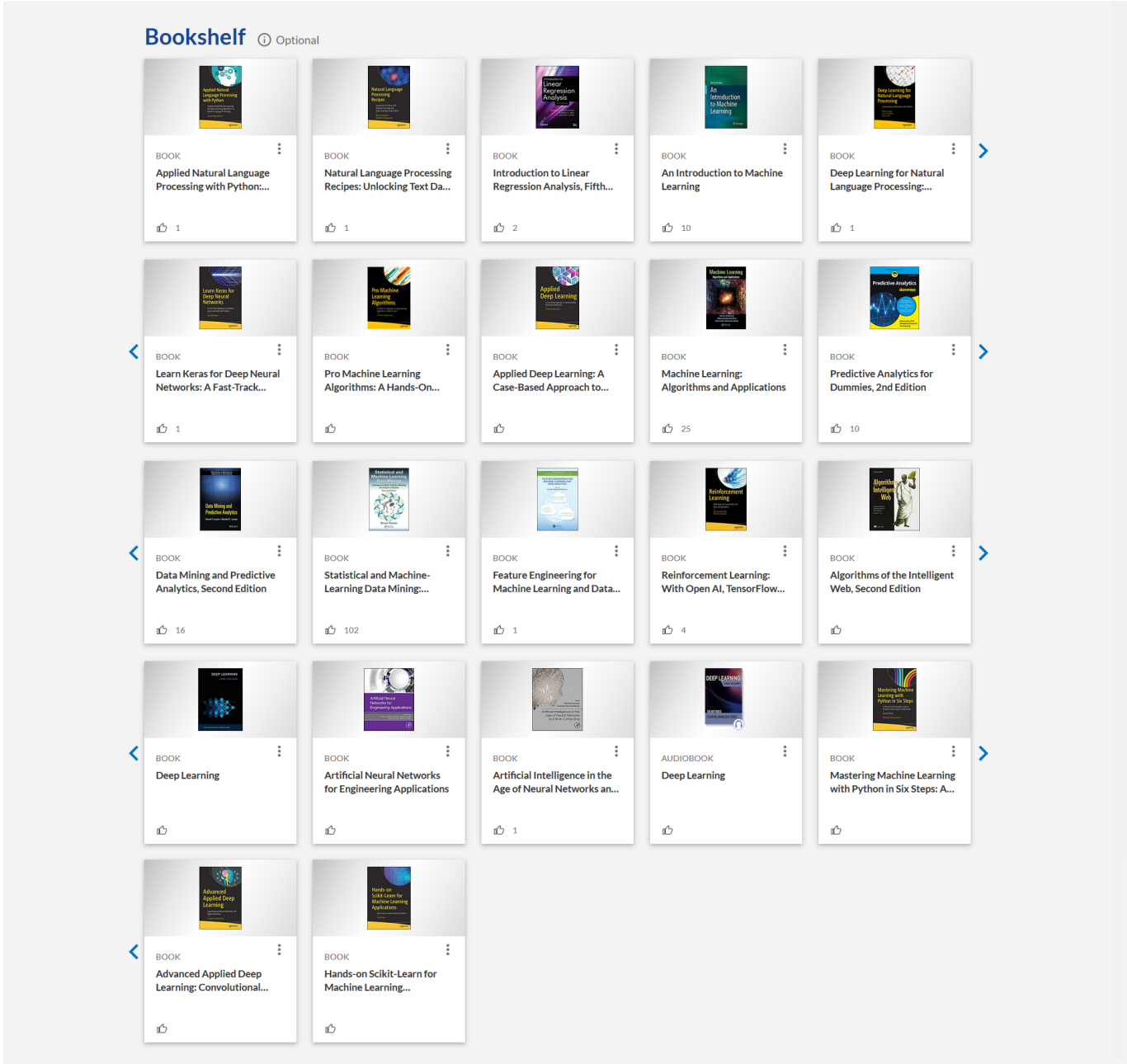
COURSE
Private Messaging & Communication Tools
9

COURSE
Creating, Finding & Sharing Information
4

COURSE
Configuring Slack
23

COURSE
Using the iOS App
3

Bookshelf for ML Architects



FOLLOW US ON:

www.skilltech.pl

email: biuro@skilltech.pl

tel. +48 22 44 88 827

SkillTech
Technology hired for excellence