Data Science
Unit 5 Chapter 1

- *** Univariate Analysis**
- Univariate analysis is the simplest form of analyzing data.
- Uni means "one," so your data has only one variable. It doesn't deal with causes or relationships, and its main purpose is to describe.
- It takes data, summarizes that data, and finds patterns in the data.
- The patterns found in univariate data include central tendency (mean, mode, and median) and dispersion, range, variance, maximum, minimum, quartiles (including the interquartile range), and standard deviation.
- Example:
- How many students are graduating with a data science degree?
- You have several options for describing data using a univariate approach.
- You can use frequency distribution tables, frequency polygons, histograms, bar charts, or pie charts.

- *** Bivariate Analysis:**
- Bivariate analysis is when two variables are analyzed together for any possible association, such as, for example, the correlation between gender and graduation with a data science degree?
- Canonical correlation in the experimental context is to take two sets of variables and see what is common between the two sets.
- Graphs that are appropriate for bivariate analysis depend on the type of variable.
- For two continuous variables, a scatterplot is a common graph.
- When one variable is categorical and the other continuous, a box plot is common, and when both are categorical, a mosaic plot is common.

- *** Multivariate Analysis:**
- Multivariate data analysis refers to any statistical technique used to analyze data that arises from more than one variable.
- This essentially models reality, in which each situation, product, or decision involves more than a single variable.
- More than two variables are analyzed together for any possible association or interactions.
- Example:
- What is the correlation between gender, country of residence, and graduation with a data science degree? Any statistical modeling exercise, such as regression, decision tree, SVM, and clustering are multivariate in nature.
- The analysis is used when more than two variables determine the final outcome.

- *** Linear Regression:**
- Linear regression is a statistical modeling technique that is used to model the relationship between an explanatory variable and a dependent variable, by fitting the observed data points on a linear equation, for example, modeling the body mass index (BMI) of individuals by using their weight.
- **1) Simple Linear Regression:**
- Linear regression is used if there is a relationship or significant association between the variables.
- This can be checked by scatterplots.
- If no linear association appears between the variables, fitting a linear regression model to the data will not provide a useful model.

- A linear regression line has equations in the following form:
- Y = a + bX,
- Where, X = explanatory variable and
- Y = dependent variable
- b = slope of the line
- a = intercept (the value of y when x = 0)
- t=0
- tMax=((300-100)/10)*((300-30)/5)
- for heightSelect in range(100,300,10):
- for weightSelect in range(30,300,5):
- height = round(heightSelect/100,3)
- weight = int(weightSelect)
- bmi = weight/(height*height)
- if bmi <= 18.5:
- BMI_Result=1
- elif bmi > 18.5 and bmi < 25:
- BMI_Result=2
- elif bmi > 25 and bmi < 30:
- BMI_Result=3
- elif bmi > 30:
- BMI_Result=4
- else:
- BMI_Result=0
- PersonLine=[('PersonID', [str(t)]),
- ('Height', [height]),
- ('Weight', [weight]),
- ('bmi', [bmi]),
- ('Indicator', [BMI_Result])]
- t+=1
- print('Row:',t,'of',tMax)
- if t==1:
- PersonFrame = pd.DataFrame.from_items(PersonLine)
- else:
- PersonRow = pd.DataFrame.from_items(PersonLine)
- PersonFrame = PersonFrame.append(PersonRow)
-
- **2) RANSAC Linear Regression:**
- RANSAC (RANdom SAmple Consensus) is an iterative algorithm for the robust estimation of parameters from a subset of inliers from the complete data set.
- An advantage of RANSAC is its ability to do robust estimation of the model parameters, i.e., it can estimate the parameters with a high degree of accuracy, even when a significant number of outliers is present in the data set.
- The process will find a solution, because it is so robust.
- Generally, this technique is used when dealing with image processing, owing to noise in the domain.
- n_samples = 1000
- n_outliers = 50
- X, y, coef = datasets.make_regression(n_samples=n_samples, n_features=1,
- n_informative=1, noise=10, coef=True, random_state=0)
- # Add outlier data

- np.random.seed(0)
- X[:n_outliers] = 3 + 0.5 * np.random.normal(size=(n_outliers, 1))
- y[:n_outliers] = -3 + 10 * np.random.normal(size=n_outliers)
- # Fit line using all data
- lr = linear_model.LinearRegression()
- lr.fit(X, y)
- # Robustly fit linear model with RANSAC algorithm
- ransac = linear_model.RANSACRegressor()
- ransac.fit(X, y)
-
- **3)Hough Transform:**
- The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing.
- The purpose of the technique is to find imperfect instances of objects within a certain class of shapes, by a voting procedure.
- This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.
- With the help of the Hough transformation, this regression improves the resolution of the RANSAC technique, which is extremely useful when using robotics and robot vision in which the robot requires the regression of the changes between two data frames or data sets to move through an environment.

- *** Logistic Regression:**
- Logistic regression is the technique to find relationships between a set of input variables and an output variable (just like any regression), but the output variable, in this case, is a binary outcome (think of 0/1 or yes/no).
- **1) Simple Logistic Regression:**
- A real-word business example would be the study of a traffic jam at a certain location in London, using a binary variable.
- The output is a categorical: yes or no. Hence, is there a traffic jam? Yes or no?
- The probability of occurrence of traffic jams can be dependent on attributes such as weather condition, day of the week and month, time of day, number of vehicles, etc.
- Using logistic regression, you can find the best-fitting model that explains the relationship between independent attributes and traffic jam occurrence rates and predicts probability of jam occurrence.
- This process is called binary logistic regression.
- The state of the traffic changes for No = Zero to Yes = One, by moving along a curve modelled.
- from sklearn import datasets, neighbors, linear_model
- Load the data.
- digits = datasets.load_digits()
- X_digits = digits.data
- y_digits = digits.target
- n_samples = len(X_digits)
- Select the train data set.
- X_train = X_digits[:int(.9 * n_samples)]
- y_train = y_digits[:int(.9 * n_samples)]
- X_test = X_digits[int(.9 * n_samples):]
- y_test = y_digits[int(.9 * n_samples):]

- Select the K-Neighbor classifier.
- knn = neighbors.KNeighborsClassifier()
- Select the logistic regression model.
- logistic = linear_model.LogisticRegression()
- Train the model to perform logistic regression.
- print('KNN score: %f' % knn.fit(X_train, y_train).score(X_test, y_test))
- Apply the trained model against the test data set.
- print('LogisticRegression score: %f'
- % logistic.fit(X_train, y_train).score(X_test, y_test))
-
- **2) Multinomial Logistic Regression:**
- Multinomial logistic regression (MLR) is a form of linear regression analysis conducted when the dependent variable is nominal with more than two levels.
- It is used to describe data and to explain the relationship between one dependent nominal variable and one or more continuous-level (interval or ratio scale) independent variables.
- You can consider the nominal variable as a variable that has no intrinsic ordering.
- This type of data is most common in the business world, as it generally covers most data entries within the data sources and directly indicates what you could expect in the average data lake.
- The data has no intrinsic order or relationship.
- You must tune a few parameters.
- t0 = time.time()
- train_samples = 5000
- Get the sample data from simulated data lake.
- mnist = fetch_mldata('MNIST original')
- Data engineer the data lake data.
- X = mnist.data.astype('float64')
- y = mnist.target
- random_state = check_random_state(0)
- permutation = random_state.permutation(X.shape[0])
- X = X[permutation]
- y = y[permutation]
- X = X.reshape((X.shape[0], -1))
- Train the data model.
- X_train, X_test, y_train, y_test = train_test_split(
- X, y, train_size=train_samples, test_size=10000)
- Apply a scaler to training to inhibit overfitting.
- scaler = StandardScaler()
- X_train = scaler.fit_transform(X_train)
- X_test = scaler.transform(X_test)
- Turn the tolerance (tol = 0.1) for faster convergence.
- clf = LogisticRegression(C=50. / train_samples,
- multi_class='multinomial',
- penalty='l2', solver='sag', tol=0.1)
- Apply the model to the data set.
- clf.fit(X_train, y_train)
- sparsity = np.mean(clf.coef_ == 0) * 100
- Score the model.
- score = clf.score(X_test, y_test)
- print('Best C % .4f' % clf.C_)

- print("Sparsity with L1 penalty: %.2f%%" % sparsity)
- print("Test score with L1 penalty: %.4f" % score)
- coef = clf.coef_.copy()

- *** Ordinal Logistic Regression:**
- Ordinal logistic regression is a type of binomial logistics regression.
- Ordinal regression is used to predict the dependent variable with ordered multiple categories and independent variables.
- In other words, it is used to facilitate the interaction of dependent variables (having multiple ordered levels) with one or more independent variables.
- This data type is an extremely good data set to process, as you already have a relationship between the data entries that is known.
- Deploying your Transform step's algorithms will give you insights into how strongly or weakly this relationship supports the data discovery process.
- **Business Problem:**
- Rate a student on a scale from 1 to 5, to determine if he or she has the requisite qualifications ("prestige") to join the university.
- The rating introduces a preference or likelihood that these items can be separated by a scale.
- StudentFrame = pd.read_csv(sFileName,header=0)
- StudentFrame.columns = ["sname", "gre", "gpa", "prestige","admit","QR","VR"
- ,"gpatrue"]
- StudentSelect=StudentFrame[["admit", "gre", "gpa", "prestige"]]
- print('Record select:',StudentSelect.shape[0])
- df=StudentSelect
- print(pd.crosstab(df['admit'], df['prestige'], rownames=['admit']))
- You should now create a rank for prestige.
- dummy_ranks = pd.get_dummies(df['prestige'], prefix='prestige')
- print(dummy_ranks.head())
- cols_to_keep = ['admit', 'gre', 'gpa']
- data = df[cols_to_keep].join(dummy_ranks.loc[:, 'prestige_1':])
- print(data.head())
- data['intercept'] = 1.00
- train_cols = data.columns[1:]
- logit = sm.Logit(data['admit'], data[train_cols])
- result = logit.fit(maxiter=500)
- print('Results')
- print(result.summary())
-
- *** Clustering Techniques:**
- Clustering (or segmentation) is a kind of unsupervised learning algorithm, in which a data set is grouped into unique, differentiated clusters.
- Let's say we have customer data spanning 1000 rows.
- Using clustering, we can group the customers into separate clusters or segments, based on the variables.
- In the case of customers' data, the variables can be demographic information or purchasing activities.
- Clustering is an unsupervised learning algorithm, because the input is unknown to the data scientist as no training set is available to pre-train a model of the solution.

- You do not train the algorithm on any past input–output information, but you let the algorithm define the output for itself.
- Therefore, there is no right solution to a clustering algorithm, only a reasonably best-fit solution, based on business usability.
- Clustering is also known as unsupervised classification.
-
- **1) Hierarchical Clustering:**
- Hierarchical clustering is a method of cluster analysis whereby you build a hierarchy of clusters.
- This works well for data sets that are complex and have distinct characteristics for separated clusters of data.
- People on a budget are more attracted to your sale items and multi-buy combinations, while more prosperous shoppers are more brand-orientated.
- These are two clearly different clusters, with poles-apart driving forces to buy an item.
- There are two design styles
- **1.Agglomerative:**
- This is a bottom-up approach. Each observation starts in its own cluster, and pairs of
- clusters are merged as one moves up the hierarchy.
- **2.Divisive:**
- This is a top-down approach. All observations start in one cluster, and splits are
- performed recursively as one moves down the hierarchy.
- **2) Partitional Clustering:**
- A partitional clustering is simply a division of the set of data objects into non-overlapping subsets (clusters), such that each data object is in exactly one subset.
- Remember when you were at school? During breaks, when you played games, you could only belong to either the blue team or the red team.
- If you forgot which team was yours, the game normally ended in disaster!
- You can generate some sample data, as follows:
- centers = [[4, 4], [-4, -4], [4, -4],[6,0],[0,0]]
- X, labels_true = make_blobs(n_samples=750, centers=centers, cluster_std=0.5,
- random_state=0)
- Let’s apply a scaler transform.
- X = StandardScaler().fit_transform(X)
- You can now apply the DBSCAN transformation.
- db = DBSCAN(eps=0.3, min_samples=10).fit(X)
- core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
- core_samples_mask[db.core_sample_indices_] = True
- labels = db.labels_
- Select the number of clusters in labels. You can ignore noise, if present.
- n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
- Here are your findings:
- print('Estimated number of clusters: %d' % n_clusters_)
- print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels_true, labels))
- print("Completeness: %0.3f" % metrics.completeness_score(labels_true, labels))
- print("V-measure: %0.3f" % metrics.v_measure_score(labels_true, labels))
- print("Adjusted Rand Index: %0.3f"
- % metrics.adjusted_rand_score(labels_true, labels))
- print("Adjusted Mutual Information: %0.3f" %
  metrics.adjusted_mutual_info_score(labels_true, labels))

- *** ANOVA:**
- The one-way analysis of variance (ANOVA) test is used to determine whether the mean of more than two groups of data sets is significantly different from each data set.
- A BOGOF (buy-one-get-one-free) campaign is executed on 5 groups of 100 customers each.
- Each group is different in terms of its demographic attributes.
- We would like to determine whether these five respond differently to the campaign.
- This would help us optimize the right campaign for the right demographic group, increase the response rate, and reduce the cost of the campaign.
- The analysis of variance works by comparing the variance between the groups to that within the group.
- The core of this technique lies in assessing whether all the groups are in fact part of one larger population or a completely different population with different characteristics.
- data.boxplot('weight', by='group', figsize=(12, 8))
- You must now perform feature extraction and engineering.
- ctrl = data['weight'][data.group == 'ctrl']
- grps = pd.unique(data.group.values)
- d_data = {grp:data['weight'][data.group == grp] for grp in grps}
- k = len(pd.unique(data.group)) # number of conditions
- N = len(data.values) # conditions times participants
- n = data.groupby('group').size()[0] #Participants in each condition
- You now need extra funtions from extra library:
- from scipy import stats
- Now activate the one-way ANOVA test for the null hypothesis that two or more
- groups have the same population mean transformation.
- F, p = stats.f_oneway(d_data['ctrl'], d_data['trt1'], d_data['trt2'])
- You need to set up a few parameters.
- DFbetween = k - 1
- DFwithin = N - k
- DFtotal = N – 1
- You can now further investigate the results from the transformation.
- SSbetween = (sum(data.groupby('group').sum()['weight']**2)/n) \ - (data['weight'].sum()**2)/N
- sum_y_squared = sum([value**2 for value in data['weight'].values])
- SSwithin = sum_y_squared - sum(data.groupby('group').sum()['weight']**2)/n
- SStotal = sum_y_squared - (data['weight'].sum()**2)/N
- MSbetween = SSbetween/DFbetween
- MSwithin = SSwithin/DFwithin
- F = MSbetween/MSwithin
- eta_sqrd = SSbetween/SStotal
- omega_sqrd = (SSbetween - (DFbetween * MSwithin))/(SStotal + MSwithin)
- Here are the results of your investigation:
- print(F,p,eta_sqrd,omega_sqrd)

- **Principal Component Analysis (PCA):**
- Dimension (variable) reduction techniques aim to reduce a data set with higher dimension to one of lower dimension, without the loss of features of information that are conveyed by the data set.
- The dimension here can be conceived as the number of variables that data sets contain.
- Two commonly used variable reduction techniques follow.

- **1) Factor Analysis:**
- The crux of PCA lies in measuring the data from the perspective of a principal component.
- A principal component of a data set is the direction with the largest variance.
- A PCA analysis involves rotating the axis of each variable to the highest Eigen vector/Eigen value pair and defining the principal components, i.e., the highest variance axis or, in other words, the direction that most defines the data.
- Principal components are uncorrelated and orthogonal.
- PCA is fundamentally a dimensionality reduction algorithm, but it is just as useful as a tool for visualization, for noise filtering, for feature extraction, and engineering.
- import matplotlib.pyplot as plt
- from mpl_toolkits.mplot3d import Axes3D
- from sklearn import datasets
- from sklearn.decomposition import PCA
- Import some data to apply your skills against.
- iris = datasets.load_iris()
- Take only the first two features.
- X = iris.data[:, :2]
- You need the following target:
- y = iris.target
- x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
- y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
- You can quickly visualize the data set.
- plt.clf()
- plt.figure(2, figsize=(8, 6))
- Plot the first three PCA dimensions.
- fig = plt.figure(1, figsize=(8, 6))
- ax = Axes3D(fig, elev=-150, azim=110)
- X_reduced = PCA(n_components=3).fit_transform(iris.data)
- ax.scatter(X_reduced[:, 0], X_reduced[:, 1], X_reduced[:, 2], c=y,
- cmap=plt.cm.Set1, edgecolor='k', s=40)
- ax.set_title("First three PCA directions")
- ax.set_xlabel("1st eigenvector")
- ax.w_xaxis.set_ticklabels([])
- ax.set_ylabel("2nd eigenvector")
- ax.w_yaxis.set_ticklabels([])
- ax.set_zlabel("3rd eigenvector")
- ax.w_zaxis.set_ticklabels([])
- plt.show()

- **Conjoint Analysis:**
- Conjoint analysis is widely used in market research to identify customers' preference for various attributes that make up a product.
- The attributes can be various features, such as size, color, usability, price, etc.
- Using conjoint (trade-off) analysis, brand managers can identify which features customers would trade off for a certain price point.
- Thus, such analysis is a highly used technique in new product design or pricing strategies.
- **Decision Trees:**
- Decision trees, as the name suggests, are a tree-shaped visual representation of routes you can follow to reach a particular decision, by laying down all options and their probability of occurrence.

- Decision trees are exceptionally easy to understand and interpret.
- At each node of the tree, one can interpret what would be the consequence of selecting that node or option.
- Before you start the example, I must discuss a common add-on algorithm to decision trees called AdaBoost.
- AdaBoost, short for "adaptive boosting," is a machine-learning meta-algorithm.
- The classifier is a meta-estimator, because it begins by fitting a classifier on the original data set and then fits additional copies of the classifier on the same data set, but where the weights of incorrectly classified instances are adjusted, such that subsequent classifiers focus more on difficult cases.
- It boosts the learning impact of less clear differences in the specific variable, by adding a progressive weight to boost the impact.

- *** Support Vector Machines, Networks, Clusters, and Grids:**
- The support vector machine (SVM) constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification and regression.
- The support vector network (SVN) daisy chains more than one SVM together to form a network.
- All the data flows through the same series of SVMs.
- The support vector cluster (SVC) runs SVM on different clusters of the data in parallel.
- Hence, not all data flows through all the SVMs.
- The support vector grid (SVG) is an SVC of an SVN or an SVN of an SVC.
- This solution is the most likely configuration you will develop at a customer site.
- It uses SVMs to handle smaller clusters of the data, to apply specific transform steps.
- **1) Support Vector Machines:**
- A support vector machine is a discriminative classifier formally defined by a separating hyperplane.
- The method calculates an optimal hyperplane with a maximum margin, to ensure it classifies the data set into separate clusters of data points.
- **2) Support Vector Networks:**
- The support vector network is the ensemble of a network of support vector machines that together classify the same data set, by using different parameters or even different kernels.
- This a common method of creating feature engineering, by creating a chain of SVMs.
- **3) Support Vector Clustering:**
- Support vector clustering is used were the data points are classified into clusters, with support vector machines performing the classification at the cluster level.
- This is commonly used in highly dimensional data sets, where the clustering creates a grouping that can then be exploited by the SVM to subdivide the data points, using different kernels and other parameters.

- *** Data Mining:**
- Data mining is processing data to pinpoint patterns and establish relationships between data entities.
- Here are a small number of critical data mining theories you need to understand about data patterns, to be successful with data mining.
- **1) Association Patterns:**
- This involves detecting patterns in which one occasion is associated to another.
- If, for example, a loading bay door is opened, it is fair to assume that a truck is loading goods.

- Pattern associations simply discover the correlation of occasions in the data.
- You will use some core statistical skills for this processing.
- Correlation is only a relationship or indication of behavior between two data sets.
- The relationship is not a cause-driven action.
- df1 = pd.DataFrame({'A': range(8), 'B': [2*i for i in range(8)]})
- df2 = pd.DataFrame({'A': range(8), 'B': [-2*i for i in range(8)]})
- print('Correlation Positive:', df1['A'].corr(df1['B']))
- print('Correlation Negative:', df2['A'].corr(df2['B']))
- df1.loc[2, 'B'] = 10
- df2.loc[2, 'B'] = -10
- print('Correlation Positive:', df1['A'].corr(df1['B']))
- print('Correlation Negative:', df2['A'].corr(df2['B']))

- **2) Classification Patterns:**
- This technique discovers new patterns in the data, to enhance the quality of the complete data set.
- Data classification is the process of consolidating data into categories, for its most effective and efficient use by the data processing.
- For example, if the data is related to the shipping department, you must then augment a label on the data that states that fact.
- A carefully planned data-classification system creates vital data structures that are easy to find and retrieve.
- You do not want to scour your complete data lake to find data every time you want to analyze a new data pattern.
-
- **Clustering Patterns:**
- Clustering is the discovery and labeling of groups of specifics not previously known.
- An example of clustering is if, when your customers buy bread and milk together on a Monday night, you group, or cluster, these customers as "start-of-the-week small-size shoppers," by simply looking at their typical basic shopping basket.
- Any combination of variables that you can use to cluster data entries into a specific group can be viewed as some form of clustering.
- For data scientists, the following clustering types are beneficial to master.
- **Connectivity-Based Clustering**
- You can discover the interaction between data items by studying the connections between them.
- This process is sometimes also described as hierarchical clustering.
- **Centroid-Based Clustering (K-Means Clustering):**
- "Centroid-based" describes the cluster as a relationship between data entries and a virtual center point in the data set. K-means clustering is the most popular centroid-based clustering algorithm.
- You now set up a data set to study.
- np.random.seed(5)
- iris = datasets.load_iris()
- X = iris.data
- y = iris.target
- Configure your estimators for the clustering.
- estimators = [('k_means_iris_8', KMeans(n_clusters=8)),
- ('k_means_iris_3', KMeans(n_clusters=3)),
- ('k_means_iris_bad_init', KMeans(n_clusters=3, n_init=1, init='random'))]

- Get ready to virtualize your results.
- fignum = 1
- titles = ['8 clusters', '3 clusters', '3 clusters, bad initialization']
- for name, est in estimators:
- fig = plt.figure(fignum, figsize=(4, 3))
- ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
- est.fit(X)
- labels = est.labels_
- ax.scatter(X[:, 3], X[:, 0], X[:, 2],
- c=labels.astype(np.float), edgecolor='k')
- ax.w_xaxis.set_ticklabels([])
- ax.w_yaxis.set_ticklabels([])
- ax.w_zaxis.set_ticklabels([])
- ax.set_xlabel('Petal width')
- ax.set_ylabel('Sepal length')
- ax.set_zlabel('Petal length')
- ax.set_title(titles[fignum - 1])
- ax.dist = 12
- fignum = fignum + 1
- **Distribution-Based Clustering**
- This type of clustering model relates data sets with statistics onto distribution models.
- The most widespread density-based clustering technique is DBSCAN.
- **Density-Based Clustering**
- In density-based clustering, an area of higher density is separated from the remainder of the data set.
- Data entries in sparse areas are placed in separate clusters.
- These clusters are considered to be noise, outliers, and border data entries.
- **Grid-Based Method**
- Grid-based approaches are common for mining large multidimensional space clusters having denser regions than their surroundings.
- The grid-based clustering approach differs from the conventional clustering algorithms in that it does not use the data points but a value space that surrounds the data points.

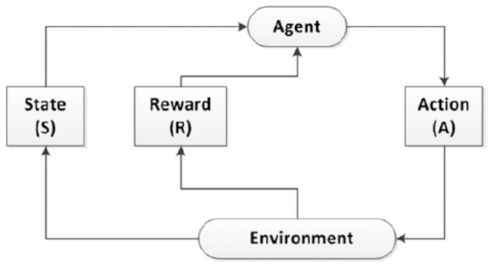
- **3) Bayesian Classification:**
- Naive Bayes (NB) classifiers are a group of probabilistic classifiers established by applying Bayes's theorem with strong independence assumptions between the features of the data set.
- There is one more specific Bayesian classification you must take note of, and it is called tree augmented naive Bayes (TAN).
- Tree augmented naive Bayes is a semi-naive Bayesian learning method.
- It relaxes the naive Bayes attribute independence assumption by assigning a tree structure, in which each attribute only depends on the class and one other attribute.
- A maximum weighted spanning tree that maximizes the likelihood of the training data is used to perform classification.
- Download the file.
- raw_data = urllib.request.urlopen(url)
- Load the CSV file into a numpy matrix.
- dataset = np.loadtxt(raw_data, delimiter=",")
- Separate the data from the target attributes in the data set.
- X = dataset[:,0:8]

- y = dataset[:,8]
- Add extra processing capacity.
- from sklearn import metrics
- from sklearn.naive_bayes import GaussianNB
- model = GaussianNB()
- model.fit(X, y)
- print(model)
- Produce predictions.
- expected = y
- predicted = model.predict(X)
- # summarize the fit of the model
- print(metrics.classification_report(expected, predicted))
- print(metrics.confusion_matrix(expected, predicted))

- **Sequence or Path Analysis:**
- This identifies patterns in which one event leads to another, later resulting in insights into the business.
- Path analysis is a chain of consecutive events that a given business entity performs during a set period, to understand behavior, in order to gain actionable insights into the data.
- This function enables you to expose all the paths between the two nodes you created
- for each customer.
- pset = nx.all_shortest_paths(G, source=NodeX, target=NodeY, weight=None)
- t=0
- for p in pset:
- t=0
- ps = 'Path: ' + str(p)
- for s in p:
- c+=1
- t+=1
- ts = 'Step: ' + str(t)
- #print(NodeX, NodeY, ps, ts, s)
- if c == 1:
- pl = [[sCustomer, ps, ts, s]]
- else:
- pl.append([sCustomer, ps, ts, s])
-
- **Forecasting:**
- This technique is used to discover patterns in data that result in practical predictions about a future result, as indicated, by predictive analytics of future probabilities and trends.
-
- **Pattern Recognition:**
- Pattern recognition identifies regularities and irregularities in data sets.
- The most common application of this is in text analysis, to find complex patterns in the data.
- grid_search = GridSearchCV(pipeline, parameters, n_jobs=-1, verbose=1)
- t0 = time()
- grid_search.fit(data.data, data.target)
- print("done in %0.3fs" % (time() - t0))
- print("Best score: %0.3f" % grid_search.best_score_)
- print("Best parameters set:")

- best_parameters = grid_search.best_estimator_.get_params()
- for param_name in sorted(parameters.keys()):
- print("\t%s: %r" % (param_name, best_parameters[param_name]))

- **Machine Learning:**
- Machine learning is the capability of systems to learn without explicit software development.
- It evolved from the study of pattern recognition and computational learning theory.
- The impact is that with the appropriate processing and skills, you can amplify your own data capabilities, by training a processing environment to accomplish massive amounts of discovery of data into actionable knowledge, while you have a cup of coffee, for example.
- This skill is an essential part of achieving major gains in shortening the data-to-knowledge cycle.
- **A. Supervised Learning:**
- Supervised learning is the machine-learning task of inferring a function from labelled training data.
- The training data consists of a set of training examples.
- In supervised learning, each example is a pair consisting of an input object and a desired output value.
- You use this when you know the required outcome for a set of input features.
- **B. Unsupervised Learning:**
- Unsupervised learning is the machine-learning task of inferring a function to describe hidden structures from unlabeled data.
- This encompasses many other techniques that seek to summarize and explain key features of the data.
- **C. Reinforcement Learning:**
- Reinforcement learning (RL) is an area of machine learning inspired by behavioural psychology that is concerned with how software agents should take actions in an environment, so as to maximize, more or less, a notion of cumulative reward.
- This is used in several different areas, such as game theory, control theory, operations research, simulation-based optimization, multi-agent systems, statistics, and genetic algorithms.



- *Reinforced learning diagram*
- Your agent extracts features from the environment that are either "state" or "reward." State features indicate that something has happened.
- Reward features indicate that something happened that has improved or worsened to the perceived gain in reward.
- The agent uses the state and reward to determine actions to change the environment.
- This process of extracting state and reward, plus responding with action, will continue until a pre-agreed end reward is achieved.

- **1) Bagging Data:**
- Bootstrap aggregating, also called bagging, is a machine-learning ensemble meta-algorithm that aims to advance the stability and accuracy of machine-learning algorithms used in statistical classification and regression.
- It decreases variance and supports systems to avoid overfitting.
- **2) Random Forests:**
- Random forests, or random decision forests, are an ensemble learning method for classification and regression that works by constructing a multitude of decision trees at training time and outputting the results the mode of the classes (classification) or mean prediction (regression) of the individual trees.
- Random decision forests correct for decision trees' habit of overfitting to their training set.
- The result is an aggregation of all the trees' results, by performing a majority vote against the range of results.
- So, if five trees return three yeses and two nos, it passes a yes out of the Transform step.
- **3) Computer Vision (CV):**
- Computer vision is a complex feature extraction area, but once you have the features exposed, it simply becomes a matrix of values.

- *** Natural Language Processing (NLP):**
- Natural language processing is the area in data science that investigates the process we as humans use to communicate with each other.
- This covers mainly written and spoken words that form bigger concepts.
- Your data science is aimed at intercepting or interacting with humans, to react to the natural language.
- There are two clear requirements in natural language processing.
- First is the direct interaction with humans, such as when you speak to your smartphone, and it responds with an appropriate answer. For example, you request "phone home," and the phone calls the number set as "home."
- The second type of interaction is taking the detailed content of the interaction and understanding its context and relationship with other text or recorded information.
- Examples of these are news reports that are examined, and common trends are found among different news reports.
- This a study of the natural language's meaning, not simply a response to a human interaction.
- **1) Text-Based:**
- The basic principle is that the library matches your text against the text stored in the data libraries and will return the correct matching text analysis.
- from nltk.tokenize import sent_tokenize, word_tokenize
- Txt = "Good Day Mr. Vermeulen,\
- how are you doing today?\
- The weather is great, and Data Science is awesome.\
- You are doing well!"
- print(Txt,'\n')
- print('Identify sentences')
- print(sent_tokenize(Txt),'\n')
- print('Identify Word')
- print(word_tokenize(Txt))
- **2) Speech-Based:**
- There is a major demand for speech-to-text conversion, to extract features.

- *** Neural Networks:**
- Neural networks (also known as artificial neural networks) are inspired by the human nervous system.
- They simulate how complex information is absorbed and processed by the human system.
- Just like humans, neural networks learn by example and are configured to a specific application.
- Neural networks are used to find patterns in extremely complex data and, thus, deliver forecasts and classify data points.
- Neural networks are usually organized in layers.
- Layers are made up of a number of interconnected "nodes."
- Patterns (features) are presented to the network via the "input layer," which communicates to one or more "hidden layers," where the actual processing is done.
- The hidden layers then link to an "output layer," where the answer is output.
- **1) Gradient Descent:**
- Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function.
- To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point.
- cur_x = 3 # The algorithm starts at x=3
- gamma = 0.01 # step size multiplier
- precision = 0.00001
- previous_step_size = cur_x
- df = lambda x: 4 * x**3 - 9 * x**2
- while previous_step_size > precision:
- prev_x = cur_x
- cur_x += -gamma * df(prev_x)
- previous_step_size = abs(cur_x - prev_x)
- print("Current X at %f" % cur_x, " with step at %f" % previous_step_size)
- print("The local minimum occurs at %f" % cur_x)
- The gradient descent can take many iterations to compute a local minimum with a required accuracy.
- This process is useful when you want to find an unknown value that fits your model.
- **2) Regularization Strength:**
- Regularization strength is the parameter that prevents overfitting of the neural network.
- The parameter enables the neural network to match the best set of weights for a general data set.
- The common name for this setting is the epsilon parameter, also known as the learning rate.

- *** TensorFlow:**
- TensorFlow is an open source software library for numerical computation using data-flow graphs.
- Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them.
- The flexible architecture allows you to deploy computation to one or more CPUs or GPUs.
- TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's machine intelligence research organization, for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains.
-

- **1) Basic TensorFlow:**
- import tensorflow as tf
- Create a TensorFlow constant.
- const = tf.constant(2.0, name="const")
- Create TensorFlow variables.
- b = tf.Variable(2.5, name='b')
- c = tf.Variable(10.0, name='c')
- You must now create the operations.
- d = tf.add(b, c, name='d')
- e = tf.add(c, const, name='e')
- a = tf.multiply(d, e, name='a')
- Next, set up the variable initialization.
- init_op = tf.global_variables_initializer()
- You can now start the session.
- with tf.Session() as sess:
- # initialise the variables
- sess.run(init_op)
- # compute the output of the graph
- a_out = sess.run(a)
- print("Variable a is {}".format(a_out))
-
- **Real-World Uses of TensorFlow:**
- TensorFlow process for the following three models:
- • Basket analysis: What do people buy? What do they buy together?
- • Forex trading: Providing recommendations on when to purchase forex for company requirements.
- • Commodities trading: Buying and selling futures.
-
- **Processing the Digits on a Container:**
- Here is a more practical application.
- You are required to identify the digits on the side of a container as it moves around your shipping yard at Hillman Ltd’s Edinburgh warehouse.
- As it rains most of the time, the scanning equipment is not producing optimum images as the containers move past the camera recording the digits.
- Can you resolve the problem of identifying the digits?
-
- The use of TensorFlow enhances your capacity in a very positive manner.
- The TensorFlow ecosystem has opened up many new opportunities for processing large and complex data sets in the cloud, returning results to the customer’s equipment.