Machine Learning

Corporate Office Address

iASYS Technology Solutions Pvt. Ltd.

25/5 Rajiv Infotech Park

Hinjewadi Phase-III Pune 411057 (India)

Ph: +91 20 2552 0602

www.iasys.co.in

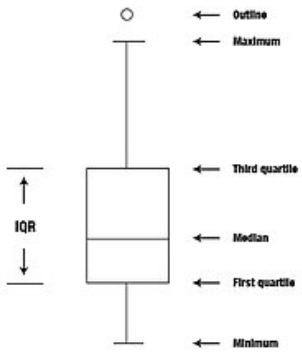
Date: *June 12th, 2023*

Documentation by: *Adharsh S*

Table of Contents

1. Basic Understanding Required

- Int – Integer: (1 // 3 // 72)
- Float: Numbers with decimals (1.03 // 3.02 // 72.5)
- String – Collection of Character (“Machine Learning”)
- List: Cluster of values with different or the same data type
- Tuple: Same as list but the values cannot be changed after the creation.
- Set: Cannot have duplicate values
- Dictionary: Key -Value pair
- Boolean: True / False
- If Elif Else
- For Loop/ While Loop - Indexing
- Functions
- Classes and Objects
  - Classes are the blueprints and objects are the instants of the class.
- Arrays (SIMD – Simple Input Multiple Data)
- Central Measure - Mean / Median / Mode
  - Mode: Categorical data
  - Mean: Numeric data with a smaller number of outliers
  - Median: Numeric data with a greater number of outliers
- Spread – Range / IQR / Standard Deviation
  - Range: Max – Min
  - IQR: 75^{th} Percentile – 25^{th} Percentile
  - SD: Sqrt (Mean of X^2 – (Mean of X) ^2)
- Visualization (EDA – Exploratory data analysis)
  - Scatter Plot
  - Histogram: For continuous data
  - Bar Plot: (X – Feature and Y – Frequency) – For discrete data
  - Pie Plot: For Discrete Data
  - Box Plot: For finding outliers.



- Data Cleaning
  - Missing / Null Values
    # Drop the data
    # Impute with the existing data
  - Data not in the right format
  - Duplicates
  - Outliers
  - Structured Data
  - Textual Data related issues
- Data Science Solutioning Process
  - Define the problem statement.
  - Collect the data for the solution.
  - Clean the data.
  - EDA (Exploratory data analysis)
  - Construct a model building (Multiple Models)
  - Validation of Models.
  - Interpret the model (Knowing how the model comes up with the conclusion)
  - Deployment

2. Machine Algorithm

Any algorithm where the rules are automatically learned by the algorithm is called the ML algorithm.

ETA Formal Definition: An algorithm is called an ML algorithm (measured using a metric A) if the performance of the algorithm increases in each task T with more experience E.

Types of Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

3. 6 JARS OF ML

3.1 Data JAR

- How to prepare the data before applying with ML
- Cleaning of the data
- Encoding of the data
  - Ordinal data (with some order) – Label Encoding (Binary category)
  - Nominal data (without some order) – One hot Encoding
- Split the data.
  - Train data – Develop the ML model.
  - Test data – Test the ML model.
- Scaling of the dataset (not mandatory)
  - Normalization / Standard Scaling
    For X data = (X – mean(X))/ S.D (X)

3.2 Task JAR

- Supervised Learning – Predict a target variable.
  - Regression Predictions – Predict a Continuous Value
  - Classification Predictions – Predict a Categorical Value
- Unsupervised Learning
  - No Target Value.
  - Analyze the pattern.
  - Clustering - group the data
  - Dimensionality reduction
- Reinforcement Learning
  - Feedback will be delayed when we do the predictions.

3.3 Model JAR

- A mathematical formula/ representation of an ML algorithm
- Y = mx +c (straight line equation), where m and c are the parameters of the model which we are the unknown values.
- Algorithm finds the best value of m and c to find a relationship between y and x.
- All the ML models can be explained using a mathematical model and all models will have parameters.
- Logistic regression 1/ 1+ e^ (-mx – c)

3.4 Loss JAR

- Formula to measure how far my predictions are from the target variable.
- Lower the loss, better the predictions, and vice versa.
- Eg: 1/n * summation (i = 1 to n) (yi – yi^)^2 : Mean Square error
- From the historical data, the best parameters will be found with the least loss.

3.5 Learning JAR

- Hit and Trial Method
  - Will try out all possible parameters and choose the least loss parameters.
- Gradient Descent Approach
  - Without knowing how the parameter vs loss graph is, we will use gradient descent to find the least loss.
  - Choose a random value for "m".
  - Compute slope
  - Walk along the slope in a downward direction.
  - Continue the same until you find the bottommost point.
  - We can find only the local best value.
  - GD algorithm doesn't work in all cases. The GD will work if there is only one minimum (ConveX). If there are many minima (non-convex), the GD will not work.
  - For a non-convex loss function, we calculate the least loss starting from multiple points of m.
  - For both convex and non-convex loss, we use GD.

3.6 Evaluation JAR

4. Supervised Learning

The Algorithm where the models are getting trained with a set of target values is called Supervised Learning.

It is of two types:

- Regression Algorithm - Predict a true value.
- Classification Algorithm - Predict a class.

4.1 Regression Algorithm

The algorithm, which is used to find a true value, is called the regression algorithm.

4.1.1 Linear Regression Algorithm

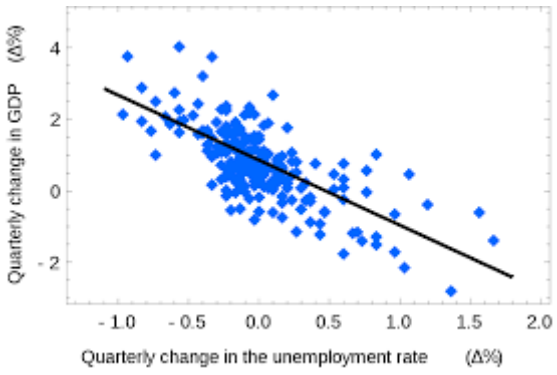
A linear regression algorithm is a model where the target and parameters are related by power 1.

Y =mx + c or Y = mx^2 + c follows linear expression.

Y = (m^2)*x + c or Y = mx +c^2 does not follow linear expression

It finds a linear relationship or straight-line relationship between the feature and target.

- Data – Clean, Encode, Split, Scale.
- Task – Regression - Supervised Learning – Predict a continuous value.
- Model – Y = mx + c, where m and c are the parameters

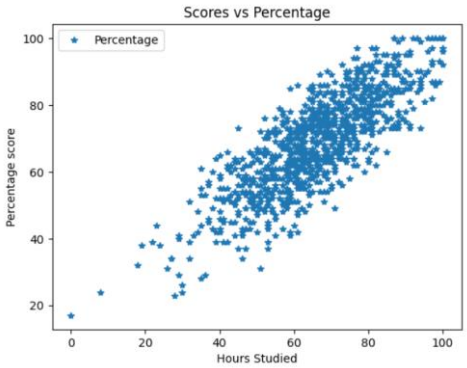


- Can be found using plotting.
- Correlation: A formula that can measure the strength of a linear relationship
- Correlation can take a value between -1 to +1
- Correlation = Summation (i= to n) [(Xi – mean(X))(Yi – mean(Y)]/[V(x)*V(y)]
- Value greater than 0.2 to 1 or -0.2 to -1, shows a good linear relationship either in positive or negative direction.
- Low correlation (-0.2 to 0.2) means no linear relationship exists. But it doesn't mean that there is no relationship.
- If there is no correlation for a linear relationship, I can transform the feature.
- X will be converted as X^2 or X^3 or sqrt(X) or e^X or log(X)
- If it results in a linear relationship, I can apply linear regression.

- Loss
  - Mean Squared error – when there is less outlier as the square is more sensitive for outliers. (Convex loss function)
  - Mean absolute error – when there are a lot of outliers. (We will consider a convex loss function)
- Learning
  - Gradient Descent
- Evaluation JAR
  - R^2 will be based on the problem statement, which is linear regression here.
  - R^2 = 1 – ((summation i=1 to n)[(Yi – Yi^)^2)/((summation i=1 to n)(Yi – Y^)^2]
  - Yi – true value, Yi^ - predicted value, Y^ - Mean of true target value)
  - R2 will take a value between – infinity to 1.
  - R2 having a higher value is better than the model.
  - If R^2 comes less than the baseline value 0 is a useless model.

Simple Linear Regression ML Model

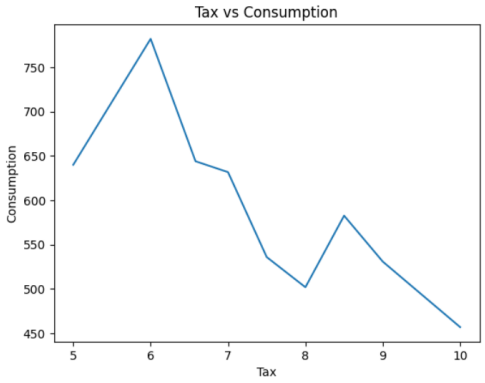
https://github.com/Adharsh0001/Machine-Learning/blob/main/Linear_Regression.ipynb




```
[ ]  1 from sklearn.linear_model import LinearRegression
     2 regressor = LinearRegression()
     3 regressor.fit(X_train,Y_train)
```

Multiple Linear Regression ML Model

https://github.com/Adharsh0001/Machine-Learning/blob/main/Multiple_Linear_Regression.ipynb




```
1 from sklearn.linear_model import LinearRegression
2 regressor = LinearRegression()
3 regressor.fit(X_train_scaled,Y_train)
```

4.1.2 Lasso and Ridge Regression

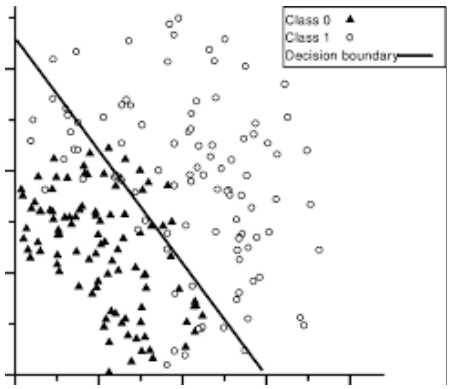
- Linear Regression becomes complex and overfitting when the features used for analysis (B0 + B1X1 +B2X2+.....+BnXn), Bi values become too less or too high.
- Hence to avoid this, we use Lasso and Ridge Regression
- Lasso (L1 Regression) = MSE + 1/n (summation of i=1 to n) (modulus (Bi)) - Convex
- Ridge (L2 Regression) = MSE + 1/n (summation of i=1 to n) (Bi^2) – Non-Convex.

4.2 Classification Algorithm

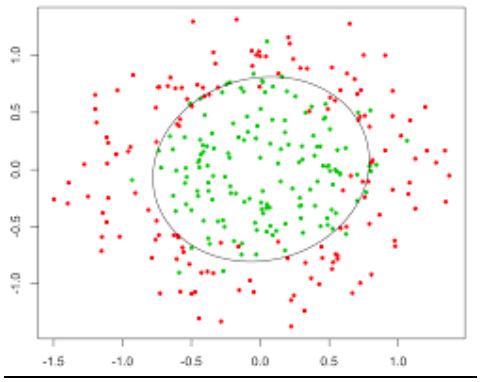
As an ML expert, I will try Logistic Regression, KNN and Decision Tree and will use the model with the best result (cross-validation score)

4.2.1 Logistic Regression

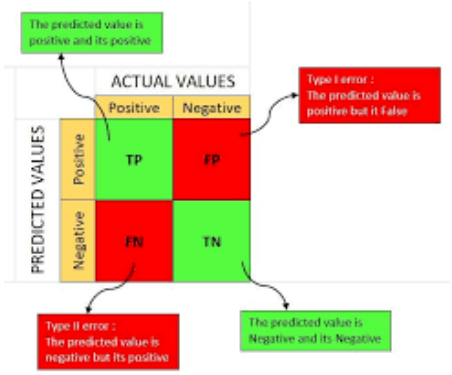
- Scaling is not mandatory for logistic regression.
- Data JAR – Clean, Encode, Split, Scale
- Task: Supervised Learning _ Logistic for classification
- Model JAR
  - Y = 1/ 1 + e^ (-mx-c) – Sigmoid Function gives graph like S
  - Y will take a value between 0 to 1
  - We will consider the threshold value as 0.5.
  - If the value goes above 0.5, we accept or if the value comes below 0.5, we reject.
- Logistic Regression model can find only linear decision boundaries.



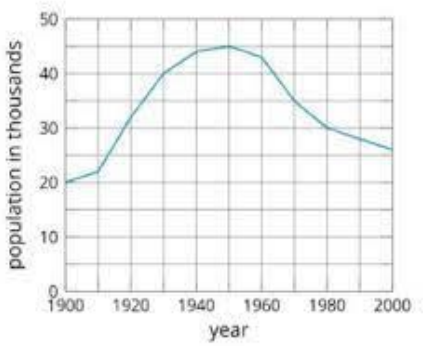
- Eg for Non-Linear Decision Boundary



- Loss JAR
  - Log Loss = summation of i =1 to n (-Yi log(Yi^))-((1-Yi)*log(1-Yi^))
  - Yi – true category, Yi^ - predicted probability
  - Lower the loss, the better the model.
- Learning JAR
  - Hit and Trial – Time-consuming
  - Gradient Descent – It will work 100% for the Logistic Regression model.
- Evaluation Metric Jar
  - Accuracy
    = (number of correct predictions/ total number of predictions) *100
  - F1-Score Confusion Matrix



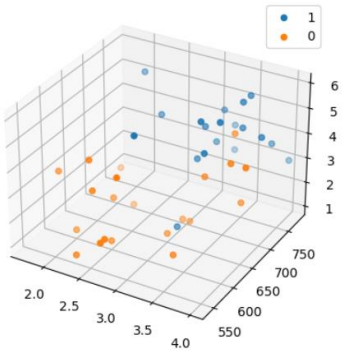
- F1 -Score = True Positive / (TP + ½ (FP +FN))
- F1 score can take values between 0 to 1.
- The higher the value, the higher the accuracy.
  - AUROC - Area under the receiver operating characteristic curve
- ROC plot
- From the confusion matrix
  True positive rate (TPR) = TP / (TP +FN)
  False positive rate (FPR)= FP / (TN+FP)
- From the TPR and FPR, we need to draw a graph which is called ROC. Where FPR will be on X-axis and TPR will be on Y-axis.
- We will get one TPR and FPR for one threshold value. So multiple threshold values are used to find more TPR and FPR.
- We will change the thresholds from 0.1 to 1 at an increment rate of 0.1.



- The area under the ROC is AVROC
- AUROC can take a value between 0 to 1.
- The higher the value, the better the model.
- AUROC value of more than 0.5 is a good model.
- AUROC value less than 0.5 is a useless model. (Added advantage over F1 score)

Logistics Regression ML Model

https://github.com/Adharsh0001/Machine-Learning/blob/main/Logistic_Regression.ipynb




```
[ ]   1 from sklearn.linear_model import LogisticRegression
      2 regressor = LogisticRegression()
      3 regressor.fit(X_train,Y_train)
      4 Y_pred = regressor.predict(X_test)
      5 Y_pred
```

4.2.2 K-Nearest Neighbour (KNN)

- For the models where we don't have linear decision boundaries, we can go with KNN's nearest neighbor.
- For classification problems, we can use KNN.
- KNN works using distance metrics.
  - Euclidian metrics
    Sqrt [(X2- X1) ^2 + (Y2 – Y1) ^2]
- It will get the K value from the datasets.
- It will compute the distance between all data points.
- Choose the K nearest neighbors.
- Predict the majority class.
- Whenever we give a value in ML, those are called hyperparameters as the user will load the K value.
- Hyperparameters are found by Hit and Trial method followed by Cross-Validation Score.
- Data – Clean / Encode/Split /Scale
  - Scaling is a must for the KNN algorithm.
- Task JAR – Supervised Learning – Classification (Can be used for regression as well)
- Model - It will get the K value from the datasets.
- It will compute the distance between all data points.
- Choose the K nearest neighbors.
- Predict the majority class.
- Loss
  - There is no loss function as there is no parameter.
  - It is called a Lazy algorithm.
- Learning
  - No Learning
- Evaluation JAR
  - Auroc / F1 score / Accuracy.

Cross Validation

- Since we can't use test data for evaluating the model, we come up with the idea of testing the model with a cross-validation method.

Data – 70% to train and 30% to test

Data is again split into 5 or 10 bins.

Let's assume 70% of data is around 1000datas.

Then each bin will have 100datas if we are using 10 bins.

Test 1 - Bin 2 to Bin 10 will be used to create the model – Bin 1 will be used to find the AUROC value.

Test 2 - Bin 3 to Bin 1 will be used to create the model – Bin 2 will be used to find the AUROC value.

.

.

.

Test m - the test continues till the number of bins is divided.

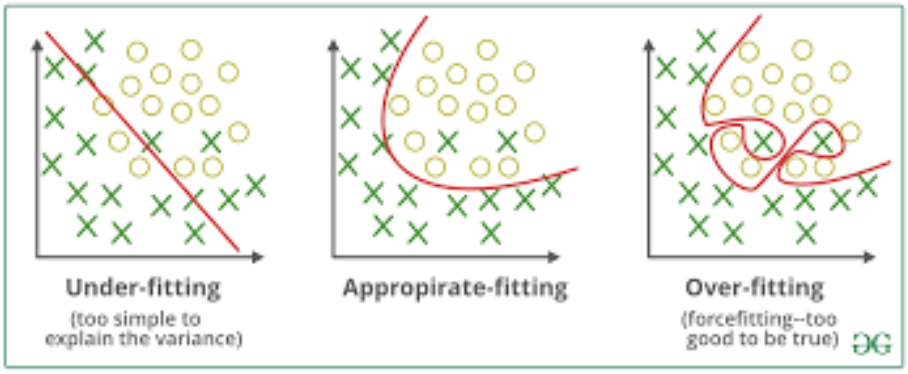
The average of all the AUROC values is considered and this process is called cross-validation.

Overfitting

- If the model starts to memorize data instead of pattern, it is called Overfitting.
- If the model is performing well with the training data than another model but performs poorer than another model over test data or validation data is called an Overfit model.

Underfit Model

- If a model is lost in the training data set and as well as in test data with another model, it is called an Underfit model.



- In a KNN model, if the K value increases, the plot decision becomes smoother i.e., the model gets smoother.
- Trade-off happens by comparing training data accuracy and cross-validation score.
- The model with the best values with both train accuracy and Val accuracy is the best model.

KNN ML Model

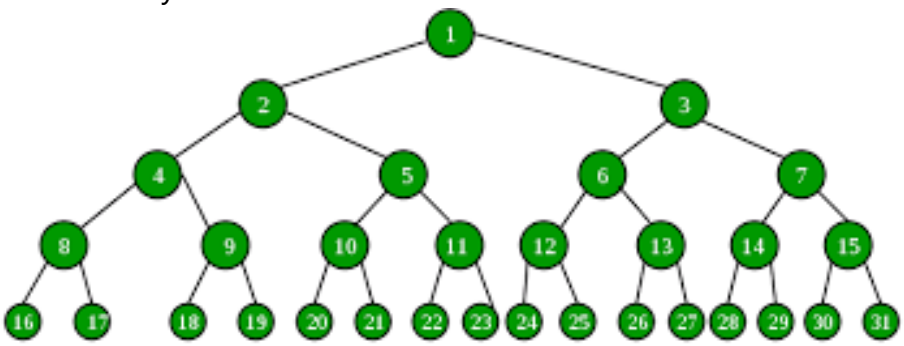
https://github.com/Adharsh0001/Machine-Learning/blob/main/KNN_Algorithm.ipynb

```
[ ]
def knn_comparison(data,k):
  X = data[["X","Y"]].values
  Y = data["Z"].astype(int).values
  X_train, X_test, Y_train,Y_test=  train_test_split(X,Y,test_size = 0.3)
  clf = KNeighborsClassifier(n_neighbors=k)
  clf.fit(X_train, Y_train)
  #print("Train Accuracy:", clf.score(X_train, Y_train))
  #print("Val Accuracy:", np.mean(cross_val_score(clf,X_train,Y_train,cv =10)))
  print("Train Accuracy:", clf.score(X_train, Y_train),"Val Accuracy:", np.mean(cross_val_score(clf,X_train,Y_train,cv =10)))
  # plot_decision_regions(X=X_train,y=Y_train,clf=clf,legend=2)
  # plt.xlabel("X")
  # plt.ylabel("Y")
  # plt.title("Knn with K=" +str(k))
  # plt.show()
```

4.2.3 Decision Trees

4.2.3.1 Decision Trees for Classification

- Decision trees can be used for both regression and classification.
- It follows the binary tree.



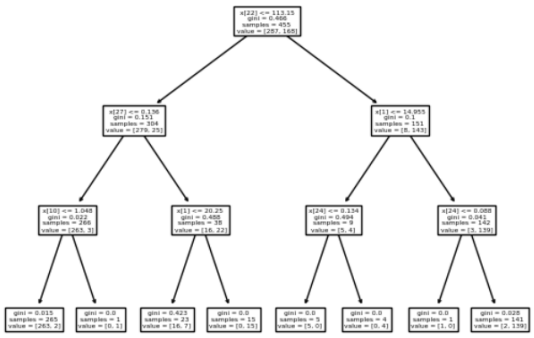
- The place where we ask questions is called a node.
- The place where we make the decision is called a leaf.
- The levels and leaves are connected by a formula (2^(n-1))
- To build a decision tree, the ML must ask the correct questions.
- A question must have a clear separation. (Right Question)
- To check the measure of the split of data, we use entropy.
  - Entropy – Measures the Randomness
  - Entropy = summation i=1 to k (-Pk * log(K))
  - K – no. of classes in the data
  - Pk – Proportion of data in class k
  - Entropy ranges between 0 to 1
  - The lower the entropy, the better the split.
  - Reduction of Entropy = entropy of the parent – the proportion from parent * entropy of child 1 - the proportion from parent * entropy of child 2
  - Reduction in entropy ranges between 0 to 1.
  - The higher the reduction, the question get selected.
- Data - Clean / Encode (Can go with Label Encode. It is sufficient) / Split / Scaling (Scaling is not mandatory for the decision tree)
- Task – Supervised Learning – Classification (Can be used for regression as well)
- Model – Follows binary tree.
- Loss – Reduction in Entropy.
- Learning – Hit and Trial
- Evaluation JAR – Accuracy/ F1-Score/ AUROC
- If the decision tree grows more, it becomes more complex; hence, we can expect to overfit the issue.

- Hence, we pass a hyperparameter that defines the depth of the decision tree.
- The hyperparameter will be finalized using the hit and trial method and cross-validation.
- The feature which has the greatest absolute value will be the best feature.

Decision Tree for Classification ML Model

https://github.com/Adharsh0001/Machine-Learning/blob/main/Decision_Tree.ipynb

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, Y_train)
Y_pred = dt.predict(X_test)
Y_pred
```



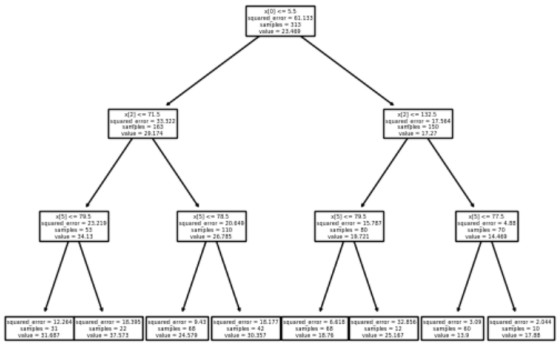
4.2.3.2 Decision Trees for Regression

- Decision tree used for Regression problems.

Decision tree for Regression ML Model

https://github.com/Adharsh0001/Machine-Learning/blob/main/Decision_Tree_for_Regression.ipynb

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(X_train, Y_train)
Y_pred = dt.predict(X_test)
Y_pred
```



4.3 Imbalanced Learning

- It's Supervised Learning.

- Imbalanced data:

Any dataset where 70% of data belongs to one class or is skewed to one class is called an Imbalanced dataset.

- Most of the classification algorithms will be based on an Imbalanced dataset.

Problems in Imbalanced datasets

Problem 1

- Accuracy is a bad metric for an Imbalanced dataset. Since even 99% of accuracy will not be sufficient for the evaluation of the dataset.
- E.g.: Fraud transaction – 1% and Proper transaction – 99%
- F1 score and AUROC can be used.

Problem 2

- All the traditional ML algorithms will give biased results for imbalanced datasets.
- We are interested in predicting the minority class, but ML will be interested in predicting the Majority class properly.
- So, we need to balance the data.
- For balancing the data, we will remove most of the value from the majority class (Under sampling) or increase more value in the minority class (Oversampling)
- Undersampling leads to Underfit issues and Oversampling leads to Overfit issues.

Smart Way to Under Sampling

- Group the majority class into sub-class by using K-Means class.
- Remove the data around some % from all the sub-class.
- Smartly removing data by preserving the pattern in the data.
- This method is called Cluster Centroid Under Sampling.

Smart Way to Over Sampling

- Choose a data point randomly.
- Compute the 2 nearest neighbors.

- Calculate the center of the three different points and add new data.
- SMOTE – Synthetic Minority Oversampling Technique

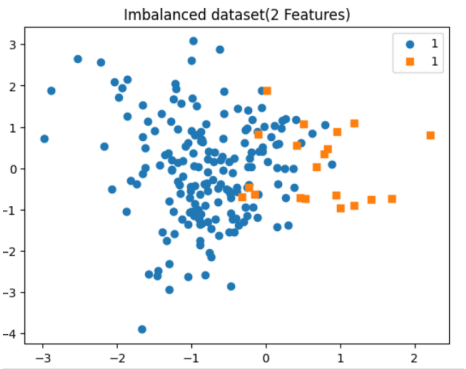
Combined Way of Balancing the Data

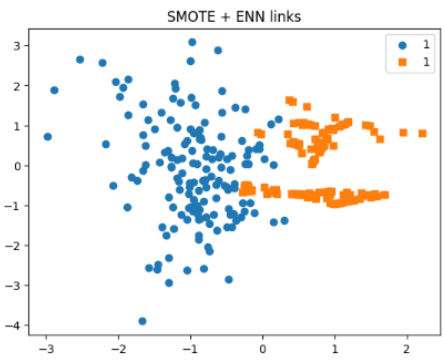
- SMOTEENN – Reduces the majority class near the Minority class and creates more minority class data to balance the dataset.
- Cluster Centroid for removing data but the data that are only close to the minority class.

Imbalanced Learning ML Model

https://github.com/Adharsh0001/Machine-Learning/blob/main/Imbalanced_Learning.ipynb

```
1 from imblearn.combine import SMOTEENN
2 plot_2d_space(X,y,"Original Data")
3 smt = SMOTEENN(sampling_strategy ='all')
4 X_smt, y_smt = smt.fit_resample(X,y)
5 plot_2d_space(X_smt,y_smt, "SMOTE + ENN links")
```



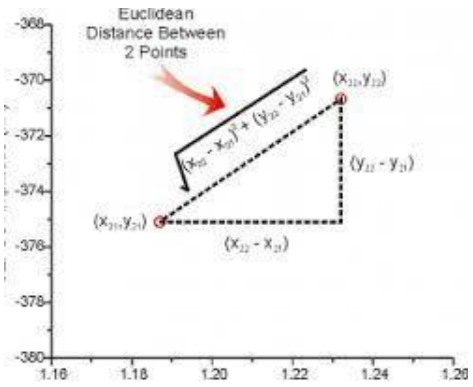


5. Unsupervised Learning

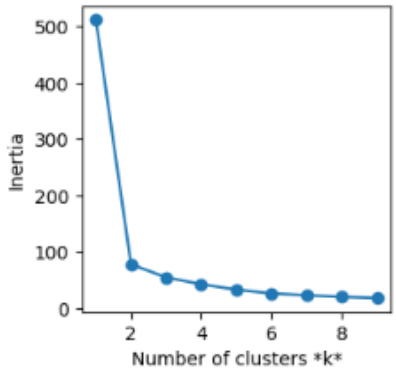
- There are no targets.
- We are interested in finding the patterns.

5.1 K – means Clustering.

- It follows Euclidian distance.



- Scaling is mandatory.
- Get the value from the data scientist.
- The K which is passed is called a hyperparameter.
- Splitting is not required as we are not trying to predict.
- Randomly chosen points (k) are called Centroids.
- Will measure the distance between each data to each centroid and will assign it to the closest to the centroids.
- Then the data will be clustered into a few groups as per the closest distance from the centroids.
- Then we will find the average value for each group and call them the centroid.
- Will repeat the process again, measuring the distance and forming a new group and finding the avg value as the new centroid.
- Will repeat this process, until the group doesn't get changed.
- To find the best value of K, we go for ELBOW PLOT



- We will have inertia for it.
- Inertia = Within the sum of squared distance/distance between two clusters.
- Inertia is the opposite of how tight a cluster is and we need a low value of inertia.
- Will plot the inertia values in a graph. The graph where it takes a sharp value, the value of k at that portion is called the best value of K.
- Randomness will affect my K-means clustering algorithm. As per the randomness, we will get different clusters every time.

Drawback of the K-Means Algorithm

- It clusters only in a spherical structure.

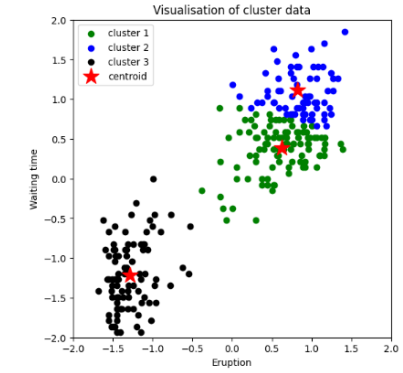
K-Means Algorithm ML Model

https://github.com/Adharsh0001/Machine-Learning/blob/main/K_means_clustering.ipynb

```
import numpy as np
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_std = scaler.fit_transform(df)

from sklearn.cluster import KMeans
#km = KMeans(n_clusters =3)
seed = np.random.randint(0,100000,size =1)[0]
km = KMeans(n_clusters=3,max_iter=3, init ="random",n_init=1,random_state=np.random.RandomState(seed))
km.fit(x_std)

centroids = km.cluster_centers_
centroids
km.labels_
km.inertia_
```


Visualisation of cluster data
cluster 1
cluster 2
cluster 3
centroid
Waiting time
Eruption
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

6. Ensemble Learning

- Collection of Multiple Machine Learning Models
- Multiple ML models will be built and the prediction of all the models will be taken for finalizing the result.
- If it is a Classification problem, we will use the mode of all the model results.
- It is called the Voting Classifier.
- If it is a Regression problem, we will use the mean of all the model results.
- It is called a Voting Regressor.
- Since the data provided for all the models are the same, the model results can be biased towards a common result. Hence to avoid this we use the Bagging concept.

Bagging: Bootstrap Aggregation

- It follows Random sampling with Replacement.
- I am going to randomly select a portion of data with replacement and pass it to each model.
- I split the data into several parts and keep it in a basket.
- I pick one part, copy the data, and return it to its basket.
- Do the process again and I get a set of data as Test Data 1
- If I repeat the process, I can create Test Data 2 and Test Data n.
- Since we use a different set of data for each model, there will be no bias.
- We are generating independent models through Bagging.
- Random Forest uses this Bagging Concept

```
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import tree
from sklearn.neighbors import KNeighborsClassfier
from sklearn.metrics import roc_auc_score

model1 = LogisticRegression(random_state =1)
model2 = tree.DecisionTreeClassifier(random_state = 1)
model3 = KNeighborsClassifer(3)
model = VotingClassifier(estimator = [("lr",model1),("dt",model2),("knn",model3)],voting ="soft")
model.fit(X_train,y_train)
preds = model.predict(X_test)
model.score(X_test,y_test)
roc_auc_score(y_test,model.predict_proba(X_test)[:,1])
```

Random Forest

- Bagging to Decision Tree
- Combination of Many decisions tree.
- By combining multiple decision trees, we will have an overfitting issue.
- Hence, we use two methods.

- Depth of each decision tree = log2(number of features)
- Will give the square root of the number of features to each model,

- Random forest can be used for regression and classification.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score

rf = RandomForestRegressor(n_estimators =100, max_depth = 3, max_features ="sqrt")
rf.fit(X_train, y_train)
predictions = rf.predict(X_test)

r2_score(predictions, y_test)
```

Boosting

- Constructs one model over another model where we build models that improve over the previous model.

Gradient Boosting

- Create Model M1 (Mean Model) which always gives the Mean value of the target variable.
- Calculate the Error between the M1 and the target variable. (Y – M1)
- Create a Model M2 using any algorithm but it takes the target variable as the error calculated for the M1 model.
- Calculate the sum of M1 and M2
- Calculate the error of M1+M2 from the true target value. (Y – M1+M2)
- Create a Model M3 using any algorithm but it takes the target variable as the error calculated for the M1+M2 model.
- Calculate the sum of M1+M2 + M3.
- Calculate the error of M1+M2+M3 from the target value.

- We create a model from the error calculated from the previous model. The model gets better and better over developing a new model over another model.

XG -Boosting (Extreme Gradient Boosting)

- All my models will be a decision tree.
- Combine M1 and M2 using M1 + Lambda M2

- Combine M1, M2 and M3 using M1 + lambda M2 + lambda^2 M3.
- Lambda is the hyperparameter.

```
[ ]  1 import xgboost as xgb
     2 from sklearn.model_selection import cross_val_score
     3 import numpy as np
     4 for lr in [0.01, 0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.2,0.5,1]:
     5   model = xgb.XGBRegressor(learning_rate =lr, n_estimators =100, verbostiy = 0)
     6   model.fit(X_train,y_train)
     7   model.score(X_test, y_test)
     8   print("Learning rate :", lr, "Train score :", modle.score(X_train, y_train),"Corss_Val_score :", np.men(corss_val_score(model,x_train)))
     9
```

Ensemble Learning ML Model

https://github.com/Adharsh0001/Machine-Learning/blob/main/Ensemble_Learning.ipynb

7. Chart

	Linear Regression	Decision Tree Regression
Data	• Clean • Encode • Split • Scaling is not mandatory but will do to get feature importance	• Clean • Encoding is not mandatory. • Split • Scaling is not mandatory
Task	Supervised Learning - Regression	Supervised Learning - Regression
Model	Y = mx+c	Follows Binary tree
Loss	MSE or MAE	Reduction in Entropy
Learning	Hit and Trial, Gradient descent	Hit and Trial
Evaluation	R2 Metrics	R2 Metrics

	Logistic Regression	KNN
Data	• Clean • Encode • Split • Scaling is not mandatory but will do to get feature importance	• Clean • Encode • Split • Scaling is mandatory
Task	Supervised Learning - Classification	Supervised Learning - Classification
Model	Y = 1/(1+e^(-mx-c))	Uses Euclidean Distance with hyperparameter K
Loss	Log Loss	No Loss Function
Learning	Hit and Trial, Gradient descent	No Learning
Evaluation	Accuracy, F1-Score, AUROC	Accuracy, F1-Score, AUROC

	Decision Tree Classification	K-Means
Data	• Clean • Encoding is not mandatory. • Split • Scaling is not mandatory	• Clean • Encode • Scaling is mandatory
Task	Supervised Learning - Regression	Unsupervised Learning -Clustering. Clustering using Centroids
Model	Follows Binary tree	Elbow Plot using Inertia
Loss	Reduction in Entropy	No loss function
Learning	Hit and Trial	No Learning
Evaluation	Accuracy, F1-Score, AUROC	No Evaluation step

8. Reference Books

INTRODUCTION TO STATISTICAL LEARNING

ELEMENTS OF STATISTICAL LEARNING

PATTERN RECOGNITION