Biobytes Project

Group 2 : Lavina,Tejas,Kshitiz

Mentors : Nischay and Vamsi

INDEX

Week 1 :

We learned python ,git and numpy and did an assignment on python and numpy and we also read two articles about role of machine learning in biotechnology

Week 2 :

We learned about pandas ,matplotlib and seaborn and did an assignment on matplotlib and seaborn and we also learned about Linear regression and decision tree and did an assignment on it

Week 3 :

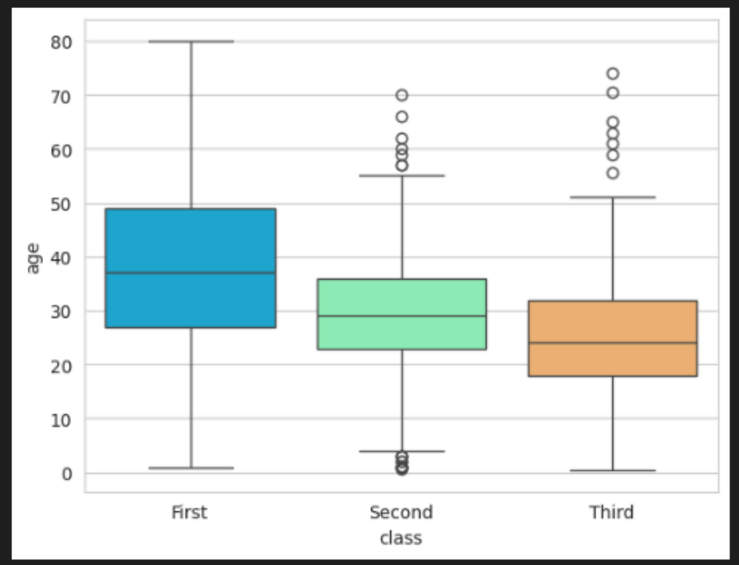
We learned about KNN and logistic regression and did an assignment on linear regression and logistic regression

Week 4 :

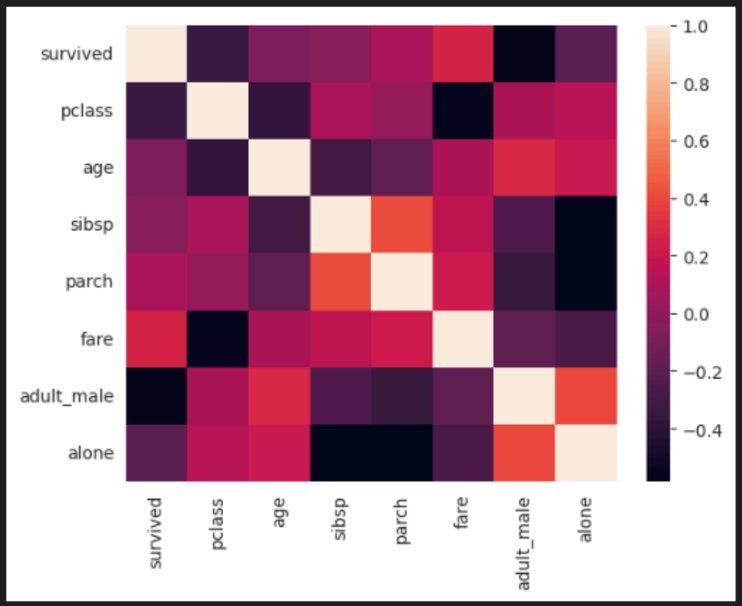
We learned about confusion matrix and categorical values

Seaborn Assignment

```
[57]  sns.boxplot(x="class", y="age", data=titanic,palette='rainbow')

...   <ipython-input-57-4d082954421f>:1: FutureWarning:

      Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the

        sns.boxplot(x="class", y="age", data=titanic,palette='rainbow')

...   <Axes: xlabel='class', ylabel='age'>
```




```
[76]  titanic.head()
      d=['sex','embarked','class','who','deck','embark_town','alive']
      df2 = titanic.drop(d, axis=1)
      df2.head()
      sns.heatmap(df2.corr())

...   <Axes: >
```



Example of boxplot and heatmap

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

[10]

```
# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)
print("Linear Regression Predictions:")
print(lr_predictions[:5])
print("Actual Home Values:")
print(y_test.head())
```

[11]

```
Linear Regression Predictions:
[[223510.2066822 ]
 [158716.72640451]
 [110365.37244064]
 [101868.55862943]
 [141959.11840445]]
Actual Home Values:
      SalePrice
258      231500
267      179500
288      122000
649       84500
1233     142000
```

```
# Decision Tree
dt_model = DecisionTreeRegressor(random_state=1)
dt_model.fit(X_train, y_train)
dt_predictions = dt_model.predict(X_test)
print("Decision Tree Predictions:")
print(dt_predictions[:5])
print("Actual Home Values:")
print(y_test.head())
```

[12]

```
Decision Tree Predictions:
[186500. 184000. 130000.  92000. 164500.]
```

ML Assignment

Linear Regression and Decision Tree Regressor predictions

Linear Regression Assignment

Read in the Ecommerce Customers csv file as a DataFrame called customers.

```
[2] customers=pd.read_csv('Ecommerce Customers')
```

Check the head of customers, and check out its info() and describe() methods.

```
[3] customers.head()
```

	Email	Address	Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	37.110597	4.104543	487.547505
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	36.721283	3.120179	581.852344
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	37.536653	4.446308	599.406092

```
[4] customers.describe()
```

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	33.053194	12.052488	37.060445	3.533462	499.314038
std	0.992563	0.994216	1.010489	0.999278	79.314782
min	29.532429	8.508152	33.913847	0.269901	256.670582
25%	32.341822	11.388153	36.349257	2.930450	445.038277
50%	33.082008	11.983231	37.069367	3.533975	498.887875
75%	33.711985	12.753850	37.716432	4.126502	549.313828
max	36.139662	15.126994	40.005182	6.922689	765.518462

Create an instance of a LinearRegression() model named lm.

```
[28] lm= LinearRegression()
```

Train/fit lm on the training data.

```
[29] lm.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

Print out the coefficients of the model

```
[30] print("Coefficients:", lm.coef_)
```

```
Coefficients: [25.98154972 38.59015875  0.19040528 61.27909654]
```

Predicting Test Data

Now that we have fit our model, let's evaluate its performance by predicting off the test values!

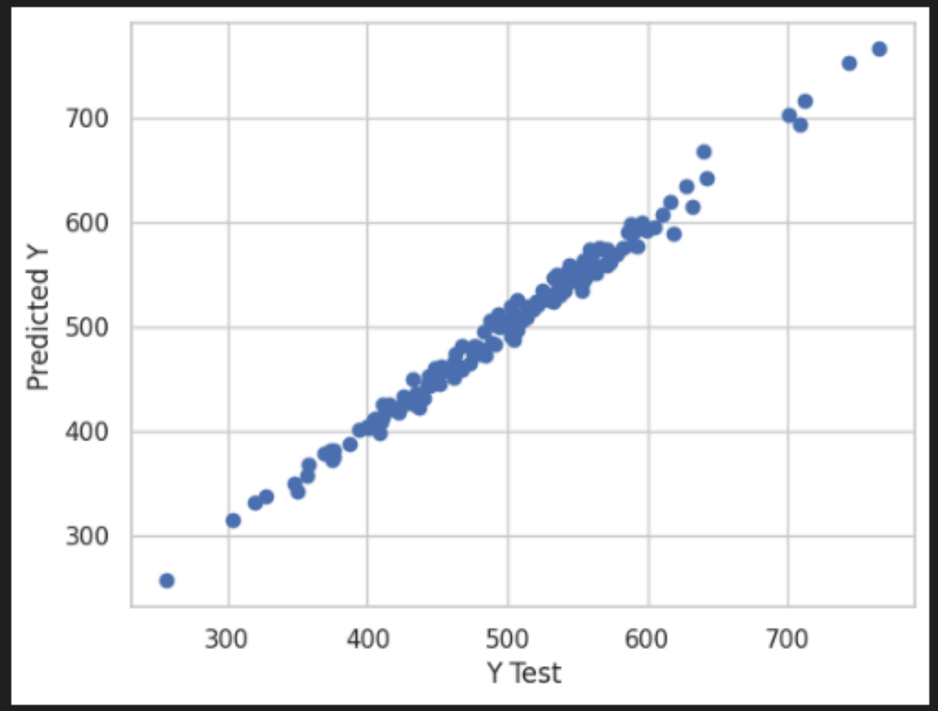
Use lm.predict() to predict off the X_test set of the data.

```
[31] # Use the trained model to make predictions on the test set
     predictions = lm.predict(X_test)

     # Display the first few predictions
     print(predictions[:10])
```

```
[456.44186104 402.72005312 409.2531539  591.4310343  590.01437275
 548.82396607 577.59737969 715.44428115 473.7893446  545.9211364 ]
```

Create a scatterplot of the real test values versus the predicted values.

```
[32] plt.scatter(y_test, predictions)
     plt.xlabel("Y Test")
     plt.ylabel("Predicted Y")
     plt.show()
```



Linear Regression model prediction

Logistic Regression Assignment

Read in the advertising.csv file and set it to a data frame called ad_data.

[38]
```
ad_data = pd.read_csv('advertising.csv')
```

Check the head of ad_data

+ Code + Markdown

[39]
```
ad_data.head()
```

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicked on Ad
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11	0
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02	0
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42	0
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19	0
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18	0

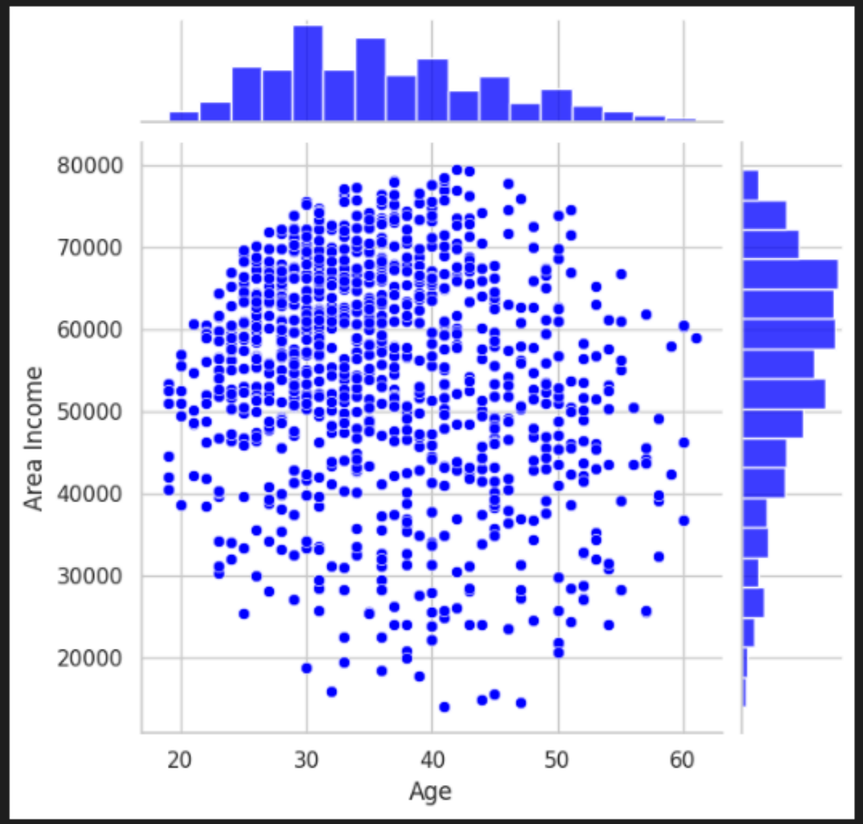
Use info and describe() on ad_data

[40]
```
ad_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Daily Time Spent on Site  1000 non-null   float64
 1   Age                       1000 non-null   int64
 2   Area Income               1000 non-null   float64
 3   Daily Internet Usage      1000 non-null   float64
 4   Ad Topic Line             1000 non-null   object
 5   City                      1000 non-null   object
 6   Male                      1000 non-null   int64
 7   Country                   1000 non-null   object
 8   Timestamp                 1000 non-null   object
 9   Clicked on Ad             1000 non-null   int64
```

Create a jointplot showing Area Income versus Age.

```
[44]
sns.set(style="whitegrid")
sns.jointplot(x='Age', y='Area Income', data=ad_data, kind='scatter', color='blue')
plt.show()
```



Train and fit a logistic regression model on the training set.

```
[50]
model = LogisticRegression()
```

```
[51]
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression()
```

Predictions and Evaluations

Now predict values for the testing data.

```
[52]
pred = model.predict(X_test)
```

Create a classification report for the model.

```
[53]
report = classification_report(y_test, pred)
```

```
[54]
# Print the classification report
print("Classification Report:\n", report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.94      0.92       105
           1       0.93      0.88      0.91        95

    accuracy                           0.92       200
   macro avg       0.92      0.91      0.91       200
weighted avg       0.92      0.92      0.91       200
```

0 21 0

Logistic Regression model prediction

kNN


Y-Axis
New example
to classify
Class A
Class B
X-Axis
Class A
Class B
Class A
Class B

Introduction to kNN

What is k-Nearest Neighbors (kNN)?

- It is used for classification and regression tasks.

- kNN is a simple, non-parametric, and lazy algorithm that classifies a data point based on the majority class among its k nearest neighbors.

.

How kNN Works?

Steps Involved in kNN :

1. Choose the number of neighbors (k):
   - k is a user-defined constant.(THE ONLY HYPERPARAMETER)

3. Calculate the distance:
   - Use a distance metric (e.g., Euclidean distance) to measure the distance between the query point and all training samples.

5. Find the nearest neighbors:
   - Identify the k points in the training data that are closest to the query point.

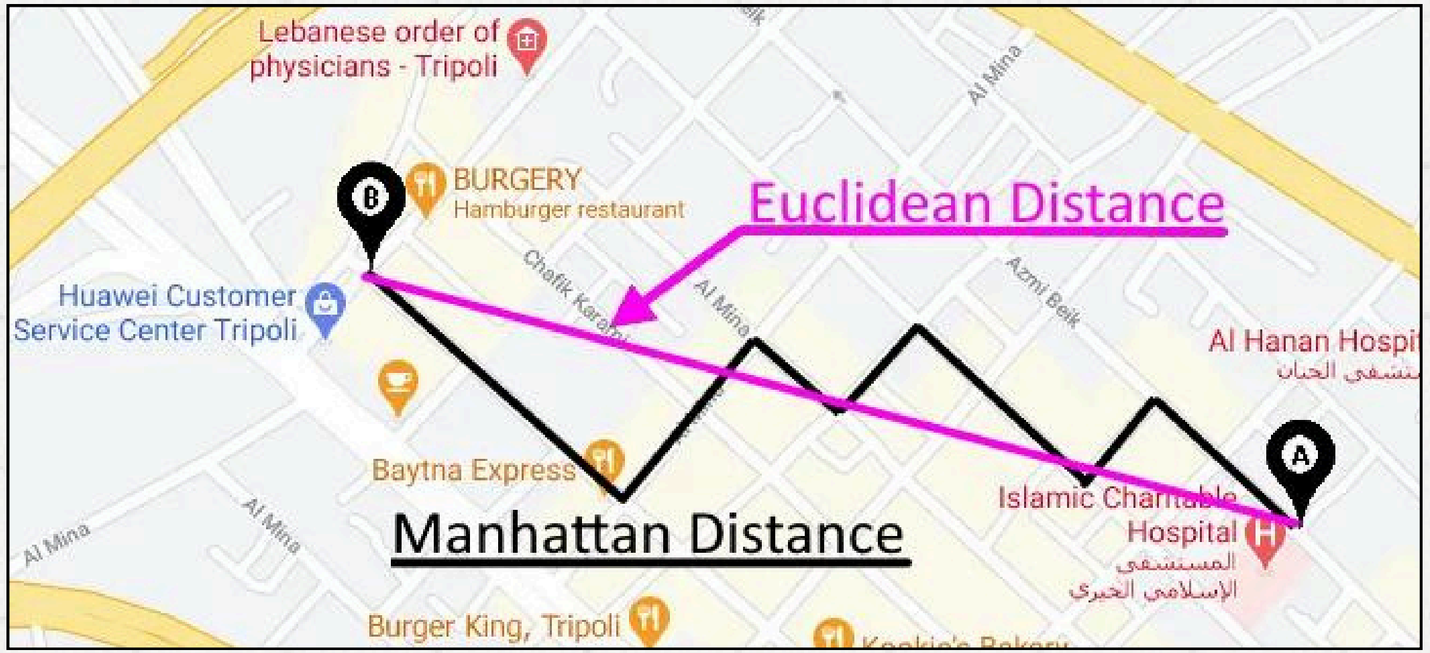
6. Make a prediction:
   - For classification: Assign the most common class among the k nearest neighbors.
   - For regression: Average the values of the k nearest neighbors.

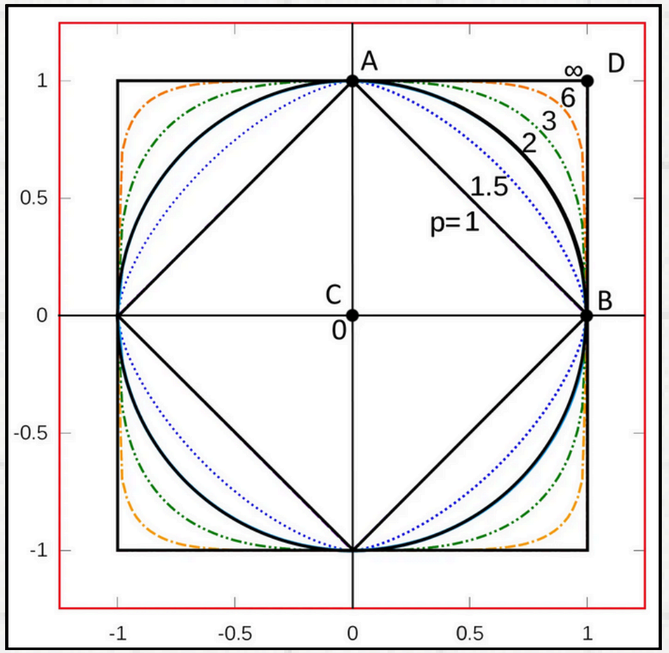
Distance Metrics :

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Manhattan Distance = $d(x,y) = \left(\sum_{i=1}^{m}\left|x_i - y_i\right|\right)$

Minkowski Distance = $\left(\sum_{i=1}^{n}\left|x_i - y_i\right|\right)^{1/p}$

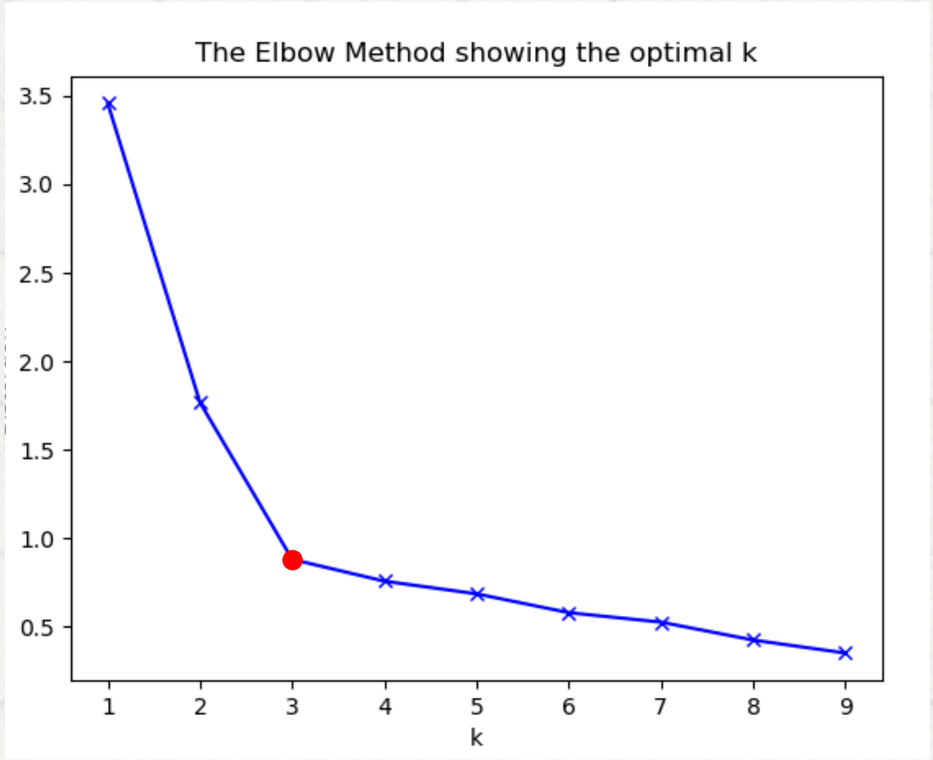




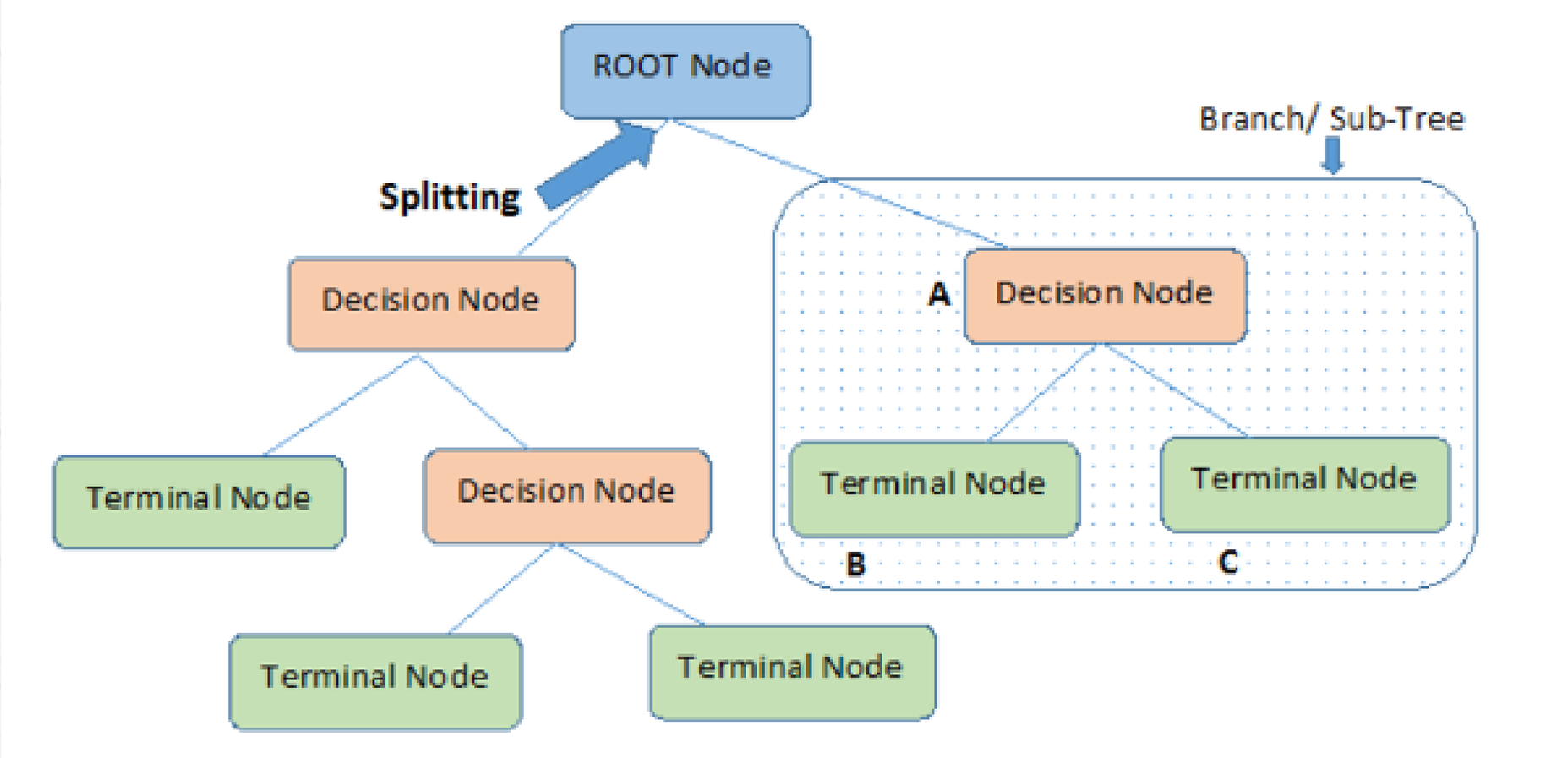
Finding Optimal Value of k :

1. Elbow Method:
   - Plot the cross-validated accuracy (or another performance metric) against different values of k. The plot typically forms an elbow shape where the error rate decreases rapidly up to a point and then starts to decrease slowly. The k at the "elbow" point is often considered optimal.

The Red Point in the figure shows the elbow



Decision Tree



Introduction to Decision Tree

What is Decision Tree?

• Decision tree is a supervised learning algorithm for classification and regression tasks.

• A decision tree is a hierarchical model used in decision support that depicts decisions and their potential outcomes.

• This algorithmic model utilizes conditional control statements, it checks if the condition is true and if it is then it goes to the next node attached to that decision

Terminologies

Root Node: The initial node at the beginning of a decision tree, where the entire population or dataset starts dividing based on various features or conditions.

Decision Nodes: Nodes resulting from the splitting of root nodes are known as decision nodes. These nodes represent intermediate decisions or conditions within the tree.

Leaf Nodes: Nodes where further splitting is not possible, often indicating the final classification or outcome. Leaf nodes are also referred to as terminal nodes.

Sub-Tree: Similar to a subsection of a graph being called a sub-graph, a sub-section of a decision tree is referred to as a sub-tree. It represents a specific portion of the decision tree.

How Decision Tree Algorithm works?

Starting at the Root: The algorithm begins at the top, called the “root node,” representing the entire dataset.

Asking the Best Questions: It looks for the most important feature or question that splits the data into the most distinct groups. This is like asking a question at a fork in the tree.

Branching Out: Based on the answer to that question, it divides the data into smaller subsets, creating new branches. Each branch represents a possible route through the tree.

Repeating the Process: The algorithm continues asking questions and splitting the data at each branch until it reaches the final “leaf nodes,” representing the predicted outcomes or classifications.

ENTROPY:

Entropy is nothing but the uncertainty in our dataset or measure of disorder.

$$E(S) = -p_{(+)}\log p_{(+)} - p_{(-)}\log p_{(-)}$$

p+ is the probability of positive class

p– is the probability of negative class

S is the subset of the training example

Information Gain :

Information gain measures the reduction of uncertainty given some feature and it is also a deciding factor for which attribute should be selected as a decision node or root node.

$$Information\ Gain = E(Y) - E(Y|X)$$

It is just entropy of the full dataset – entropy of the dataset given some feature.

Thank you !