

## **Bank Marketing with Machine Learning Approach**

**DS8015 (Machine Learning Non-DS Student)**

**Urmi Patel**

**501064008**

**Computer Science Department**

**Ryerson University**

**Jay Harshadbhai Patel**

**501138837**

**Electrical & Engineering Department**

**Ryerson University**

### **Contents**

a) Problem Definition .....	2
b) Data Description .....	2
b1) Attributes Description .....	2
b2) Statistics of the data (used tools learnt in this course to generate data statistics) .....	4
b3) Which attributes are used, which ones are eliminated? Why? .....	4
b4) How did you clean-up the data?.....	5
c) Work Distribution.....	8
d) Solution Description .....	8
Model 1: Decision Tree .....	9
Model 2: Naïve Bayes .....	10
Model 3: KNN (K-Nearest Neighbors) .....	11
Model 4: Logistic Regression .....	12
Model 5: MLP (Multi-Layer Perceptron).....	13
e) Predications deducted from the data.....	15
f) Future Work.....	17
g) References .....	18
Topics Covered from Lectures .....	20

## **a) Problem Definition**

Machine learning is widely used to build a prediction model using various datasets that can provide better accuracy. The data analysis techniques are beneficial for business and marketing purposes. The aim of this project is to build a machine learning model that helps to produce a better prediction. However, the raw dataset is converting into effective decision-making knowledge. Building the predictive models will help to predict whether the client will subscribe for a term deposit or not!

Moreover, the bank can decide doing proactive marketing instead of mass marketing and because of that, the bank can save time, costs and improve the quality of work. Bank marketing dataset is built based on the phone calls which are done by the bank for a marketing campaign.

## **b) Data Description**

From the UCI website, a popular dataset for bank marketing is used for the project. The dataset link provided as below: <https://archive.ics.uci.edu/ml/datasets/bank+marketing>

This dataset has information about the Portuguese banking institution including personal and financial information of the bank's clients. There were two datasets available of which we chose bank-additional-full.csv dataset. Each record has twenty descriptive explanations about the client and one observation of a bank term deposit would be yes or no.

Although the dataset has no missing values, it does have outliers. The dataset includes a total of 21 attributes with different data types such as numerical, categorical, and binary. Moreover, the csv file has 41188 number of instances (rows) [1].

### **b1) Attributes Description**

The detailed description of each attribute is written in the table below, where the left column has the original feature name in the dataset, the middle column is its description, and the right column has data types of that feature. The "y" attribute is the response called the desired target.

Attributes	Description	Data Types
Age	Client's age	Numerical
Job	Client's job type	Categorical
Marital	Client's marital status	Categorical
Education	What level of education client have	Categorical
Default	Has credit in default?	Categorical
Housing	Does it have a housing loan or not?	Categorical
Loan	Has a personal loan or not?	Categorical
Contact	Contact communication type	Categorical
Month	Last contract month of year	Categorical
Day_of_week	Day of the week of last contact with client	Categorical
Duration	Last contact duration in seconds	Numerical
Campaign	Number of contacts with the client during the campaign	Numerical
Pdays	Number of days that passed by after the client was last contacted from a previous campaign	Numerical
Previous	Number of contacts with the client before the campaign	Numerical
Poutcome	Outcome of the previous marketing campaign	Categorical
Emp.var.rate	Quarterly indicator of employment variation rate	Numerical
Cons.price.idx	Monthly indicator of consumer price index	Numerical
Cons.conf.idx	Monthly indicator of confidence price index	Numerical
Euribor3m	Daily indicator of Euribor 3 month rate	Numerical
Nr.employed	Quarterly indicator of number of employees	Numerical
y	Has the client subscribed a term deposit?	Binary

Table 1: Dataset Attribute Descriptions

## b2) Statistics of the data (used tools learnt in this course to generate data statistics)

```
bank_df.describe()
```

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567583	962.475454	0.172963	0.081886	93.575564	-40.502600	3.621291	5167.035911
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198	1.734447	72.251528
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.800000
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	96.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.049000	5228.100000

Figure 1: statistic of banking dataset

Pandas library have a method called describe (), which returns the descriptive statistics of the columns where count shows total number of not null values, mean for mean of the values, std for standard deviation of the observation, min and max for minimum and maximum of the values in the object and 25,50 and 75 are the default percentile, because parameters inside the method were not provided [2].

**b3) Which attributes are used, which ones are eliminated? Why?**

In case of attribute selection, we need to drop some columns which we will not use for our predictive models. Here we dropped a total of four columns from the dataset such as duration, contact, month, and day\_of\_weak. We are building a model that predicts the category of people who are most likely to subscribe to the term deposit and for that, we need the attributes that are identified before contact. Therefore, parameters related to customer information are considered as independent variables.

Moreover, we dropped the y column because we need to separate the target variable y from the dataframe. The below figure shows there are a total of 16 columns available after the attribute selection phase. This dataframe is further used for the model's training purpose.

```
#selection of independent variable. We are building the model that predict the category
newfinal_df = newfinal_df.drop(["duration", "contact", "month", "day_of_week", 'y'], axis=1)
newfinal_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 16 columns):
#   Column                Non-Null Count  Dtype
---  -
0   age                   41188 non-null  int64
1   job                   41188 non-null  int64
2   marital               41188 non-null  int64
3   education             41188 non-null  int64
4   default               41188 non-null  int64
5   housing               41188 non-null  int64
6   loan                  41188 non-null  int64
7   campaign              41188 non-null  int64
8   pdays                41188 non-null  int64
9   previous              41188 non-null  int64
10  poutcome              41188 non-null  int64
11  emp.var.rate          41188 non-null  float64
12  cons.price.idx        41188 non-null  float64
13  cons.conf.idx         41188 non-null  float64
14  euribor3m             41188 non-null  float64
15  nr.employed           41188 non-null  float64
dtypes: float64(5), int64(11)
memory usage: 5.0 MB
```

Figure 2: New dataframe after dropping columns

#### b4) How did you clean-up the data?

Data cleaning includes cleaning of the dataset's missing or null values, removing duplicate values, and converting the categorical type of variables into a machine-readable format. The missing or null values can be a significant problem in the data analysis phase because it affects the resulting accuracy of the applied models. Therefore, missing values should be removed before applying any modelling techniques on the dataset.

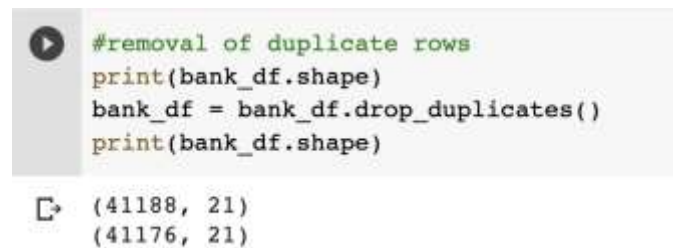
```
bank_df.isnull().sum()
```

```
age                0
job                0
marital            0
education          0
default            0
housing            0
loan               0
contact            0
month              0
day_of_week        0
duration           0
campaign           0
pdays            0
previous           0
poutcome           0
emp.var.rate       0
cons.price.idx     0
cons.conf.idx      0
euribor3m          0
nr.employed        0
y                  0
dtype: int64
```

Figure 3: Checking missing values from the dataset

Above result shows there are no missing values inside the banking marketing dataset.

An important part of the data analysis phase is analyzing duplicate values and removing them from the dataset. Duplicate data affects customer service, performs poor analysis and takes up pointless data storage space.



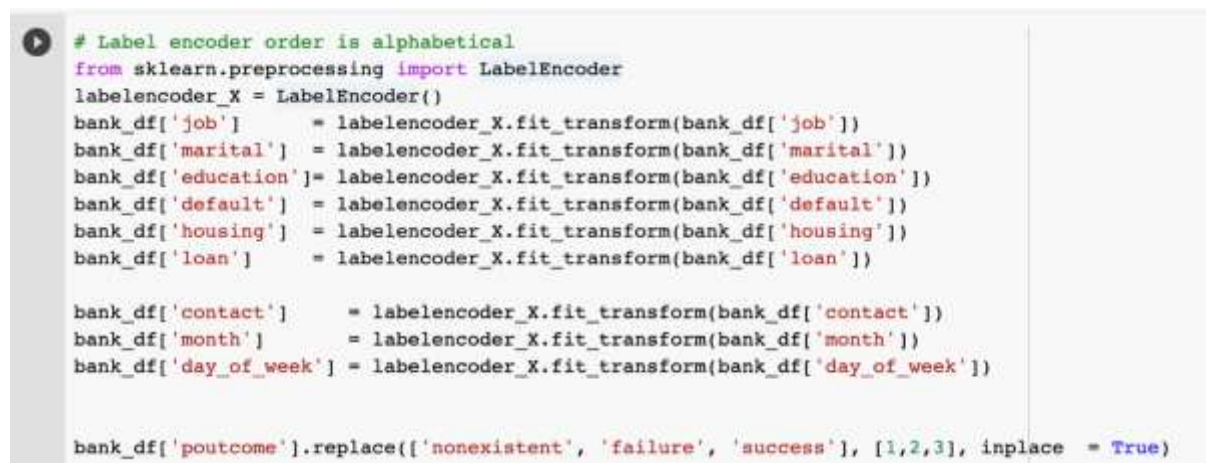
```
#removal of duplicate rows
print(bank_df.shape)
bank_df = bank_df.drop_duplicates()
print(bank_df.shape)
```

```
(41188, 21)
(41176, 21)
```

Figure 4: Removing of duplicate rows

There are 12 total duplicate rows which are available in the dataset, which is removed from the dataframe. Now, there are 41176 rows available in the dataset.

There are many categorical types of data available in the dataset such as job, outcome, month, loan and so on. Here the label encoding method is used to convert this data into machine-readable format. This method is used to replace the categorical value with a numeric value starting with 0. For this technique we used sklearn library and imported the Label Encoder.



```
# Label encoder order is alphabetical
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()

bank_df['job'] = labelencoder_X.fit_transform(bank_df['job'])
bank_df['marital'] = labelencoder_X.fit_transform(bank_df['marital'])
bank_df['education'] = labelencoder_X.fit_transform(bank_df['education'])
bank_df['default'] = labelencoder_X.fit_transform(bank_df['default'])
bank_df['housing'] = labelencoder_X.fit_transform(bank_df['housing'])
bank_df['loan'] = labelencoder_X.fit_transform(bank_df['loan'])

bank_df['contact'] = labelencoder_X.fit_transform(bank_df['contact'])
bank_df['month'] = labelencoder_X.fit_transform(bank_df['month'])
bank_df['day_of_week'] = labelencoder_X.fit_transform(bank_df['day_of_week'])

bank_df['poutcome'].replace(['nonexistent', 'failure', 'success'], [1,2,3], inplace = True)
```

Figure 5: Label encoding

Firstly, create an instance of the LabelEncoder() method and store it in a label encoder object. Secondly, use fit technique and transform the categorical data into numerical data. We used this numerical data to perform the machine learning tasks.

```
#converted age into group category
def age(df):
    df.loc[df['age'] <= 35, 'age'] = 1
    df.loc[(df['age'] > 35) & (df['age'] <= 45), 'age'] = 2
    df.loc[(df['age'] > 45) & (df['age'] <= 70), 'age'] = 3
    df.loc[(df['age'] > 70), 'age'] = 4

    return df

age(bank_df);

# convert duration in some parts and label those
def duration(df):
    df.loc[df['duration'] <= 100, 'duration'] = 1
    df.loc[(df['duration'] > 100) & (df['duration'] <= 175), 'duration'] = 2
    df.loc[(df['duration'] > 175) & (df['duration'] <= 325), 'duration'] = 3
    df.loc[(df['duration'] > 325) & (df['duration'] <= 640), 'duration'] = 4
    df.loc[df['duration'] > 640, 'duration'] = 5

    return df

duration(bank_df);
```

Figure 6: Age and duration column labeling

For the age column, we converted the values into four groups such as below age 35 it shows group 1 and the age group between 35 to 45 it shows group 2 and so on. On the other hand, the duration column also split into 5 various groups with some conditions.

Index	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	outcome	emp.var.rate	cons.price.idx	cons.conf.idx
0	3	3	1	0	0	0	0	1	8	1	3	1	999	0	1	1.1	93.994	-36.4
1	3	7	1	3	1	0	0	1	8	1	2	1	999	0	1	1.1	93.994	-36.4
2	2	7	1	3	0	2	0	1	8	1	3	1	999	0	1	1.1	93.994	-36.4
3	2	0	1	1	0	0	0	1	8	1	2	1	999	0	1	1.1	93.994	-36.4
4	3	7	1	3	0	0	2	1	8	1	3	1	999	0	1	1.1	93.994	-36.4
5	2	7	1	2	1	0	0	1	8	1	3	1	999	0	1	1.1	93.994	-36.4
6	3	0	1	5	0	0	0	1	8	1	2	1	999	0	1	1.1	93.994	-36.4
7	2	1	1	7	1	0	0	1	8	1	3	1	999	0	1	1.1	93.994	-36.4
8	1	9	2	5	0	2	0	1	8	1	4	1	999	0	1	1.1	93.994	-36.4
9	1	7	2	3	0	2	0	1	8	1	1	1	999	0	1	1.1	93.994	-36.4
10	2	1	1	7	1	0	0	1	8	1	1	1	999	0	1	1.1	93.994	-36.4
11	1	7	2	3	0	2	0	1	8	1	3	1	999	0	1	1.1	93.994	-36.4
12	1	1	2	3	0	0	2	1	8	1	2	1	999	0	1	1.1	93.994	-36.4
13	3	3	0	0	0	2	0	1	8	1	3	1	999	0	1	1.1	93.994	-36.4
14	1	1	1	1	0	2	0	1	8	1	2	1	999	0	1	1.1	93.994	-36.4
15	3	5	1	2	1	2	2	1	8	1	2	1	999	0	1	1.1	93.994	-36.4

Figure 7: Final output after conversions

The above picture shows the result after the conversions and we can see the age, job and education columns are now showing numerical values. Moreover, dependent variables are converted into dummy variables because we need the target variable(y) for our prediction which is already in the yes or no (categorical) type of data. Dummy encoding converts the categorical data into binary data.

### c) Work Distribution

Performed data analysis and data cleaning steps together via zoom meeting. Split the model building task in half and evaluate the results. In another meeting, the results were discussed, and the individual experiences shared. Perform model comparison and final analysis task as a team. Started writing the report and making PPT on google Docs and Slides.

Team 2	Urmi (member 1)	Jay (member 2)
Task 1	Data understanding and Data cleaning / preprocessing (via zoom meeting)	
Task 2	Model 1 and 2	Model 3 and 4
Task 3	Prediction 1	Prediction 2
Task 4	Model Comparison and result evaluation (via zoom meeting)	
Task 5	Report writing and ppt making (via Google's platform)	

Table 2: Teamwork distribution

### d) Solution Description

After this project we can categorize or identify a group of customers who are most likely to give a positive response on similar campaign offers, and this result will help the bank for future work and marketing strategies.

Last step of the data preparation is splitting the dataset into two parts, training, and testing. The training dataset is used to train the model and the testing dataset is used to evaluate the model's performance. The size of the divided dataset is 80:20 ratio, respectively. For splitting the dataset, we used sklearn library. Parameters such as test size = 20%, both the data frames as inputs and random\_state = 101 applied to the function. Random state is used for shuffling the applied data [3].

On the other hand, the k fold method is used to split the data into k numbers of subsets where k subsets are used for the test set and k-1 subsets used for the training set. We define a total of three parameters with kfold function such as 10 number of splits, shuffle is true and random state is zero [4].



```
[26] from sklearn.model_selection import train_test_split
      from sklearn.model_selection import KFold
      from sklearn.model_selection import cross_val_score
      from sklearn.metrics import confusion_matrix, accuracy_score

      X_train, X_test, y_train, y_test = train_test_split(newfinal_df, y, test_size = 0.20, random_state = 101)
      k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
```

Figure 8: splitting of the dataset

## Model 1: Decision Tree

### 1.1 Libraries

For this method, the sklearn library is used where the DecisionTreeClassifier is imported.

### 1.2 Why were they chosen for that particular problem space

The reason for choosing this model is, it can handle the categorical data very well. In our case, the classification tree is used for categorical types of predicted variables (y) such as yes and no or 0 and 1.

### 1.3 How they were used and explaining the logic behind the code snippets

Decision tree is simple to implement, and it follows a flow chart type structure to make a decision. Firstly, the classifier object is created and after that, the decision tree classifier is trained using a fit function. In the end, predict the response for the test dataset.

### 1.4 Code snippets and explanation

```
#Model 2 Decision Tree
from sklearn.tree import DecisionTreeClassifier

dt_model = DecisionTreeClassifier(criterion='gini').fit(X_train, y_train)
pred_DT = dt_model.predict(X_test)
print('Confusion Matrix:')
print(confusion_matrix(y_test, pred_DT))
DTree_CV = cross_val_score(dt_model, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean()
print('Accuracy for Decision Tree:', round(accuracy_score(y_test, pred_DT), 2)*100)

Confusion Matrix:
[[6630  628]
 [ 669 309]]
Accuracy for Decision Tree: 84.0
```

Figure 9: Decision tree model

From the above code, we get the accuracy result 84% for the decision tree model. Moreover, we also display the confusion matrix which is a summary of the predicted result. The confusion matrix compares the actual target values with those predicted by the machine learning model where columns represented actual values and rows represented predicted values [5] [6].

Cross-validation is a resampling method that uses various portions of the data to perform testing on the model and also train the model on the various iterations. This method is mostly

used while performing prediction tasks. On the other hand, cross-validation is helpful to check how accurately a model will perform. There are few parameters used for the cross-validation such as the object of the model where we fit the data, training data, testing data, cv , n\_jobs and scoring. The cv parameter is used to specify the number of folds we want and we use 10 folds. Scoring use for a scorer callable object which is accuracy and n\_jobs = 1 uses for the number of jobs run in parallel manner.

### 1.5 Justification of the chosen model parameters

For the decision tree we chose criterion = gini where the criterion function is used to measure the quality of a split and gini is used for Gini impurity. We did not define max\_depth of the tree, hence as a default it will be expanded until all leaves are pure.

## Model 2: Naïve Bayes

### 2.1 Libraries

For this method the sklearn library is used where the GaussianNB is imported to perform the task.

### 2.2 Why they were chosen for that particular problem space

The reason for choosing this model is, it works well with a large scale of dataset and needs less time for the training process [7].

### 2.3 How they were used and explaining the logic behind the code snippets

Firstly, the classifier object is created to train a Gaussian Naive Bayes classifier on the training set and fit the model. In the end, predict the response for the test dataset [8].

### 2.4 Code snippets and explanation

```
#Model 1 Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
GNB_model = GaussianNB().fit(X_train, y_train)
Predt_GNB = GNB_model.predict(X_test)
print('Confusion Matrix:')
print(confusion_matrix(y_test, Predt_GNB))
print('Accuracy for Gaussian NB:', round(accuracy_score(y_test, Predt_GNB), 2)*100)
GNB_CV = (cross_val_score(GNB_model, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())

Confusion Matrix:
[[6384  874]
 [ 494 484]]
Accuracy for Gaussian NB: 83.0
```

Figure 10: Gaussian NB model

From the above code we get the accuracy result 83% for the gaussian NB model. Moreover, we applied the same cross-validation method for the result.

## 2.5 Justification of the chosen model parameters

For the GaussianNB model we did not define any parameter as it will take a default. There are two parameters named `priors = None`, and `var_smoothing = 1e-09` where smoothing is used for calculation stability. After applying default parameters still, it gave comparatively good accuracy result [9].

## Model 3: KNN (K-Nearest Neighbors)

The KNN algorithm assumes similar things near to each other. This algorithm has no explicit training steps because all the work happens during the prediction phase.

### 3.1 Libraries

For this method the sklearn library is used where the `KNeighborsClassifier` is imported to perform the task.

### 3.2 Why they were chosen for that particular problem space

KNN works based on the local minimum of the target function which is used to learn an unknown function of desired precision and accuracy. KNN also finds the neighborhood of an unknown input, its distance from it, and other parameters. This algorithm works based on the principle of information gain and because of that it finds out which is most suitable to predict an unknown value or any test input [10].

### 3.3 How they were used and explaining the logic behind the code snippets

Firstly, we need to select the `k` number of neighbours which is very important when analyzing the dataset to avoid overfitting. The classifier object is created and fit the data into the model. At last, we got the accuracy for our trained model.

### 3.4 Code snippets and explanation

```
#Model 3 KNN
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

k = 40
model_KNN = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
Pred_KNN = model_KNN.predict(X_test)
print("For k=40, Accuracy for KNN: ", round(metrics.accuracy_score(y_test, Pred_KNN),4)*100,"%")
print('Confusion Matrix:')
print(confusion_matrix(y_test, Pred_KNN))
KNNModel_CV = {cross_val_score(model_KNN, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean()

For k=40, Accuracy for KNN: 89.72 %
Confusion Matrix:
[[ 7174   84]
 [ 763 215]]
```

Figure 11: KNN model

From the above code we get the accuracy result 89.72% for KNN model which is quite high compared to previous models. We obtained the confusion matrix for this model and also applied the cross-validation method for the accuracy result.

### 3.5 Justification of the chosen model parameters

```
#Model 3 KNN
from sklearn import model_selection
from sklearn.neighbors import KNeighborsClassifier
X_train1, X_test1, y_train1, y_test1 = train_test_split(newfinal_df, y, test_size = 0.2, random_state = 101)

knn_acr = []
from sklearn import metrics
for i in range(1,50):
    KNN_nr = KNeighborsClassifier(n_neighbors = i).fit(X_train1,y_train1)
    KNN_predict = KNN_nr.predict(X_test1)
    knn_acr.append(metrics.accuracy_score(y_test1, KNN_predict))

plt.figure(figsize=(10,6))
plt.plot(range(1,50),knn_acr,color = 'green',linestyle='--',
        marker='d',markerfacecolor='blue', markersize=10)
plt.title('K value vs. Accuracy')
plt.xlabel('K value')
plt.ylabel('Accuracy')
print("The highest accuracy is:-",max(knn_acr)," and the optimum number of K is=",knn_acr.index(max(knn_acr)))

The highest accuracy is:- 0.8975230694511899 and the optimum number of K is= 40
```

Figure 12: Parameter selection in KNN

We tried the k (n\_neighbors) value from 1 to 50 and we got the highest accuracy for k = 40. Hence, we continued our process with k = 40. Choosing the perfect k value for the model is the main parameter for this model. Rest of the parameters like weights and leaf size were kept as default, hence the model gave a good accuracy score [11] [12].

## Model 4: Logistic Regression

In case of classification problems, this is the most used model when the targeted variable is 0 or 1 [13].

### 4.1 Libraries

For this method the sklearn library is used where the LogisticRegression is imported.

### 4.2 Why they were chosen for that particular problem space

The main reason behind using this model is it gives a very fast result for unknown records classification. Most of the time, logistic regression gives very good accuracy results compared to other models. Moreover, it learns a linear relationship from the dataset and then introduces a non-linearity in the form of the Sigmoid function.

### 4.3 How they were used and explaining the logic behind the code snippets

Firstly, create an object for the logistic regression function. Next, fit the training data into the model and predict the result by applying the test dataset. Sigmoid function is used to map the predicted values to probabilities between 0 to 1.

#### 4.4 Code snippets and explanation

```
[33]: #Model 4 Logistic Regression
from sklearn.linear_model import LogisticRegression
logistic_model = LogisticRegression().fit(X_train,y_train)
pred_logi = logistic_model.predict(X_test)
print('Confusion Matrix:')
print(confusion_matrix(y_test, pred_logi))
print('Accuracy for Logistic Regression:',round(accuracy_score(y_test, pred_logi),2)*100)
LogiModel_CV = (cross_val_score(logistic_model, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())

Confusion Matrix:
[[7187  71]
 [ 783 195]]
Accuracy for Logistic Regression: 90.0
```

Figure 13: Logistic regression model

From the above code we get the accuracy result 90% for Logistic Regression algorithm which is the highest among all performed algorithms. Moreover, we performed the confusion matrix and cross-validation method to get the results.

#### 4.5 Justification of the chosen model parameters.

In the case of parameters, we tried different values for random\_state but we got the best result by just considering the default value for this parameter. Model considers the value of C = 1.0 and solver = lbfgs by default where the solver uses it for the optimization problem. Moreover, max\_iter parameter is used for the maximum number of iterations used for the converging process and by default it takes 100 iterations for the model.

### Model 5: MLP (Multi-Layer Perceptron)

We used the PCA (Principal Component Analysis) method to reduce the amount of time and memory required by the algorithm. PCA uses dimensionality-reduction method which means it reduces the dimensionality of large data sets, by transforming a large set of variables into a smaller one. In short, PCA can reduce the number of features to get a better analysis result. However, choosing the right number of dimensions is very important for the PCA method. For example, applying n\_components = 0.95 parameter to the PCA method means it will automatically select the number of components that makes sure that the ratio of variance is greater or equal to 95%.

```
[41] from sklearn.decomposition import PCA

# PCA method for feature reduction
pca = PCA(n_components=0.95)
pca.fit(X_train)
PCA_X_train = pca.transform(X_train)
PCA_X_test = pca.transform(X_test)
```

Figure 14: PCA method

## 5.1 Libraries

For this method the sklearn library is used where the MLPClassifier is imported.

## 5.2 Why they were chosen for that particular problem space

MLP uses multiple layers to train the model and it also uses supervised learning technique named backpropagation for the training purpose.

## 5.3 How they were used and explaining the logic behind the code snippets

MLP builds up a total of three layers called input layer, hidden layer, and output layer. Firstly, create the object ml for the model and pass parameters. Next, fit the training data to the model where the timing of the fitting data is various depending on the parameter settings. Performed the prediction method using a test dataset and obtained the accuracy result.

## 5.4 Code snippets and explanation

```
[43] from sklearn.neural_network import MLPClassifier
# define and train an MLPClassifier named ml on the given data
ml = MLPClassifier(hidden_layer_sizes=(50,100,50), max_iter=250, activation='relu', solver='adam', learning_rate_init=0.001, random_state=1)
ml.fit(PCA_X_train, y_train)

MLPClassifier(hidden_layer_sizes=(50, 100, 50), max_iter=250, random_state=1)

[48] print('Accuracy for the MLP:', ml.score(PCA_X_test, y_test))
# draw the confusion matrix
predict_ml = ml.predict(PCA_X_test)
confusion_matrix_ml = confusion_matrix(y_test, predict_ml)
print('Confusion matrix:')
print(confusion_matrix_ml)

Accuracy for the MLP: 0.8715395823215153
Confusion matrix:
[[6935  323]
 [ 735 243]]

from sklearn.metrics import accuracy_score, mean_squared_error
# print the training error and MSE
print('Training error: %f' % ml.loss_curve_[-1])
print('Accuracy:', accuracy_score(y_test, predict_ml))
print('MSE: %f' % mean_squared_error(y_test, predict_ml))

Training error: 0.159174
Accuracy: 0.8715395823215153
MSE: 0.128460
```

Figure 15: MLP model

From the above code we got 87.15% accuracy for MLP model. This model takes approximately 6 minutes to train. Moreover, obtained confusion matrix, training error and mean squared

error of the trained MLP model. Model's training error is only 15.91% and mean squared error is 12.84% which is reasonably low.

### 5.5 Justification of the chosen model parameters

There are various parameters chosen for the MLP model after some basic level of parameter tuning such as three hidden layers (50,100,50), maximum iterations = 250, activation function = relu, solver = adam, learning\_rate\_init = 0.001 and random state = 1. The solver Adam uses as a stochastic gradient-based optimizer and whenever the adam solver is used, the initial learning rate is 0.001 by default. The activation function relu (Rectified Linear Activation) is used because it does not activate all the neurons at the same time. Maximum number of iterations used in a model because the solver iterates that number of times until the convergence [18] [19].

## e) Predications deducted from the data

### Comparison of models:

If we compare the models by their accuracy, the result shows in the table below.

The highest accuracy we got for the logistic regression model. KNN model gave results comparatively very close to the logistic regression which is 89.72%. Overall, the lowest accuracy result we got from the naïve bayes model which is somewhat low, just 83%. However, the decision tree gave 84% of accuracy which is almost similar to NB.

Model Name	Accuracy
Logistic Regression	90.0 %
Gaussian NB	83.0 %
KNN	89.72 %
Decision Tree	84.0 %
MLP	87.15 %

Table 3: Accuracy comparison for all models



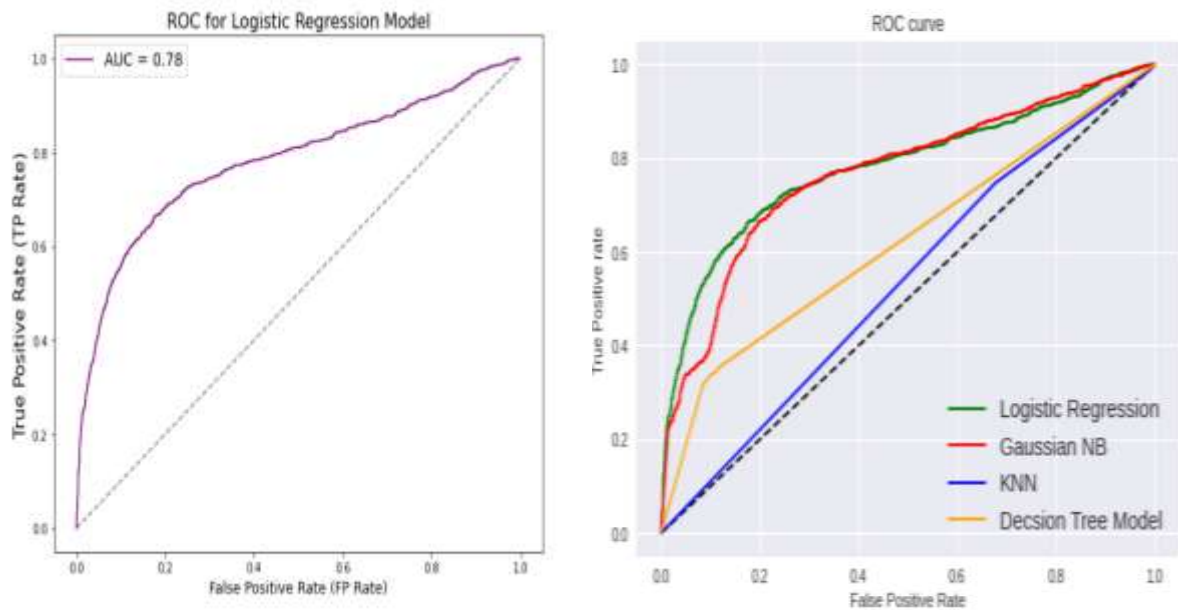


Figure 16: ROC result of models

ROC (Receiver Operating Characteristic) curve is useful when we need to predict the probability of any binary outcome. ROC is used to plot the false positive rate (on x-axis) and the true positive rate (on y-axis) within 0.0 to 1.0 threshold values [20].

Model Names	Precision		Recall		F1-Score		AUC	Accuracy (%)
	Yes (1)	No (0)	Yes (1)	No (0)	Yes (1)	No (0)		
Logistic Regression	0.73	0.90	0.20	0.99	0.31	0.94	0.78	90.0
Gaussian NB	0.36	0.93	0.49	0.88	0.41	0.90	0.77	83.0
KNN	0.72	0.90	0.22	0.99	0.34	0.94	0.53	89.72
Decision Tree	0.33	0.91	0.32	0.91	0.33	0.91	0.62	84.0

Table 4: evaluation result of the models

Above table shows overall evaluation results of the models. Table contains Precision, Recall, F1-Score, AUC and accuracy of the models. We did not include the MLP model in the table because it does not have the Precision, Recall and F1-Score measurements [21].

### Prediction 1:

For No Class (people who didn't subscribe to bank deposit), Logistic Regression performed better than other models with the highest F1 score (0.94). Moreover, for logistic regression AUC and Accuracy scores are also higher with 0.78 and 90% respectively.



**Confusion Matrix for Logistic Regression:**

7187	71
783	195

False positive means the model categorizes the customers into Yes category but they did not subscribe to the term deposit. In our opinion, a model with a smaller number of false positives is more useful. By comparing all models, we can conclude that logistic regression performed better with only 71 false positives.

**Prediction 2:**

For Yes Class (people who subscribed to bank deposit), Gaussian NB model performed with the highest F1-score.

**Confusion Matrix for Gaussian NB:**

6384	874
494	484

**f) Future Work**

There are few aspects that are planned as a future work to expand this research on a bank marketing dataset. First part is data analysis where outliers' techniques can be applied to achieve better results. Second part is a machine learning approach where more complex algorithms can be tried such as SVM, CNN, Adaboost and compare those with old work (model's results). Additionally, hyper parameter tuning for the neural networks helps to get more accurate results.

## g) References

- [1] "UCI Machine Learning Repository: Bank Marketing Data Set."  
<https://archive.ics.uci.edu/ml/datasets/bank+marketing> (accessed Apr. 23, 2022).
- [2] "pandas.DataFrame.describe — pandas 1.4.2 documentation."  
<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html>  
(accessed Apr. 23, 2022).
- [3] "scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation."  
<https://scikit-learn.org/stable/index.html> (accessed Apr. 23, 2022).
- [4] "K-Fold | K-fold Averaging on Deep Learning Classifier," *Analytics Vidhya*, Sep. 16, 2021. <https://www.analyticsvidhya.com/blog/2021/09/how-to-apply-k-fold-averaging-on-deep-learning-classifier/> (accessed Apr. 23, 2022).
- [5] "Confusion Matrix for Machine Learning," *Analytics Vidhya*, Apr. 17, 2020.  
<https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/>  
(accessed Apr. 23, 2022).
- [6] "1.10. Decision Trees," *scikit-learn*. <https://scikit-learn.org/stable/modules/tree.html>  
(accessed Apr. 23, 2022).
- [7] B. Akkaya and N. Çolakoğlu, *Comparison of Multi-class Classification Algorithms on Early Diagnosis of Heart Diseases*. 2019.
- [8] "K-Fold | K-fold Averaging on Deep Learning Classifier," *Analytics Vidhya*, Sep. 16, 2021. <https://www.analyticsvidhya.com/blog/2021/09/how-to-apply-k-fold-averaging-on-deep-learning-classifier/> (accessed Apr. 23, 2022).
- [9] "sklearn.naive\_bayes.GaussianNB," *scikit-learn*. [https://scikit-learn.org/stable/modules/generated/sklearn.naive\\_bayes.GaussianNB.html](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html) (accessed Apr. 23, 2022).
- [10] "The KNN Algorithm - Explanation, Opportunities, Limitations," *neptune.ai*, Jul. 12, 2021. <https://neptune.ai/blog/knn-algorithm-explanation-opportunities-limitations>  
(accessed Apr. 23, 2022).
- [11] "k-nearest neighbor algorithm in Python," *GeeksforGeeks*, Apr. 09, 2019.  
<https://www.geeksforgeeks.org/k-nearest-neighbor-algorithm-in-python/> (accessed Apr. 23, 2022).

- [12] "sklearn.neighbors.KNeighborsClassifier," *scikit-learn*. <https://scikit-learn/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html> (accessed Apr. 23, 2022).
- [13] S. Swaminathan, "Logistic Regression — Detailed Overview," *Medium*, Jan. 18, 2019. <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc> (accessed Apr. 23, 2022).
- [14] "Kaggle: Your Home for Data Science." <https://www.kaggle.com/> (accessed Apr. 23, 2022).
- [15] "NumPy." <https://numpy.org/> (accessed Apr. 23, 2022).
- [16] "pandas - Python Data Analysis Library." <https://pandas.pydata.org/> (accessed Apr. 23, 2022).
- [17] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decis Support Syst*, 2014, doi: 10.1016/j.dss.2014.03.001.
- [18] K. Choudhury, "Deep Neural Multilayer Perceptron (MLP) with Scikit-learn," *Medium*, Aug. 31, 2020. <https://towardsdatascience.com/deep-neural-multilayer-perceptron-mlp-with-scikit-learn-2698e77155e> (accessed Apr. 25, 2022).
- [19] "sklearn.neural\_network.MLPClassifier," *scikit-learn*. [https://scikit-learn/stable/modules/generated/sklearn.neural\\_network.MLPClassifier.html](https://scikit-learn/stable/modules/generated/sklearn.neural_network.MLPClassifier.html) (accessed Apr. 25, 2022).
- [20] "AUC-ROC Curve in Machine Learning Clearly Explained," *Analytics Vidhya*, Jun. 15, 2020. <https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/> (accessed Apr. 25, 2022).
- [21] K. P. Shung, "Accuracy, Precision, Recall or F1?," *Medium*, Apr. 10, 2020. <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9> (accessed Apr. 25, 2022).

## Topics Covered from Lectures

1	Week 4	Python Third Party Library (Pandas, Numpy, Sklearn, etc.)
2	Week 5	EDA (Pandas, Missing DATA, Categorical Data, Distribution analysis using Visualization, Descriptive Statistic)
3	Week 6	Introduction to ML (Classification Analysis, Feature selection, Model selection)
4	Week 7	End to end Machine Learning label encoding and Dummy encoding, 10-fold cross validation, ensemble methods)
5	Week 10	Classification Analysis
6	Week 11	PCA (Standardization)
7	Week 12	MLP Neural Network