Do (wo)men talk too much in films? Project in Machine Learning

Anonymous Author(s)
Number of group members: 2

Abstract

In this document, we looked at the typical 1037 films data set of Hollywood movies and their leading roles. We analyzed the data set, brought insights out of it, and tried to come up with the assumption that we could predict the lead role in a movie based on what features (given in the data set). For example, Revenue generated, year, and the number of dialogues could be one of the main features to predict the leading or main role in the films. Later we implemented Logistic Regression and Random Forest models and checked how authentic our assumptions were based on the data analysis we did earlier. We also added a naive classifier and compared it with our models. Further, using Cross Validation, we checked the accuracy of our models and came up with one model that gave us more accurate predictions. This model will be used on the test data set to check how well it performs.

1 Introduction

Film Industry has been here for a very long now, and it has had a huge impact on any country's image and contributed to its economy as a separate industry and provides a huge amount of viewership. While this industry has been here, there have been some flaws in the system, which has been lacking in keeping fair standards and equal opportunities, causing the serious issue of gender biasness. For reference, let's have a look at the graph of the Star War movie series and how gender biasness exists there, and how fewer female roles are given in all the series of Star Wars. [1].

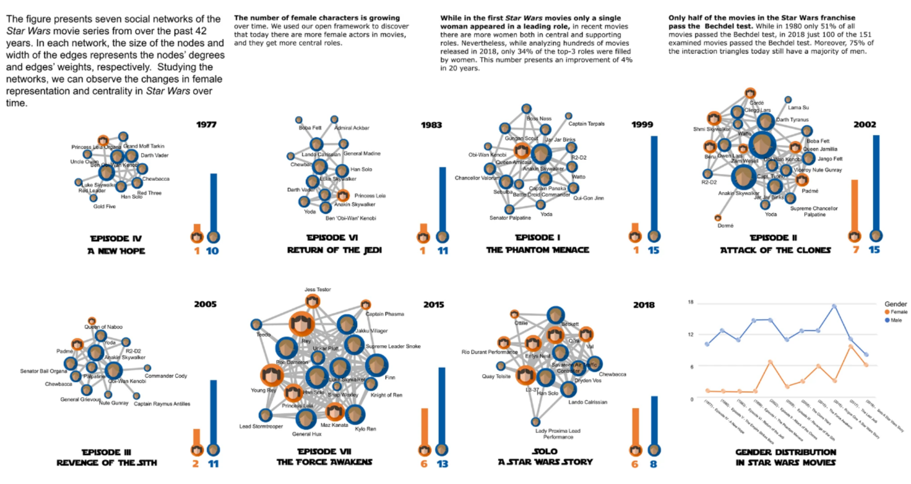

Figure 1: Star Wars Films

[1]https://www.nature.com/articles/s41599-020-0436-1/figures/1

We can clearly see in the graph above that over the past few years of the episodes of star wars, the number of female roles has been very low as compared to men, and they have been kept aside. We will be closely looking at a similar data set where we have a list of films and their characteristics, and we will be evaluating if the gender of the lead role is based on certain characteristics or features of the films. If that is true, then what are those main features, and how is it predicted? We have a data set of 1037 films and the gender of lead roles in some Hollywood movies(Male/Female). Along with the main leading roles, we have multiple features in the data set that may help us predict the gender of the leading role, e.g., the number of dialogues spoken by male actors. Female actors, the year in which the movie was released, the revenue movie created, the age of lead roles and co-lead roles, the number of male and female actors in the movie, etc. We will be analyzing this data and coming up with some key features and insights from the data set, checking if we can predict the gender of a leading role in any film.

2 Data Analysis Task

Q:1: Do men or women dominate speaking roles in Hollywood movies?
From sample data, we can see that over the years from 1939-2015, in most films, male actors have dominated compared to females. Talking about the ratio, as seen in the graph, 76% of lead roles have been covered by males, and only 24% is there for women. The training data also shows that the ratio of male and female actors working in a film is very different. In the majority of the films, the number of male actors working is very high compared to the females, thus giving male actors more probability to be the lead role. So, we can see the clear dominance of the male actors.

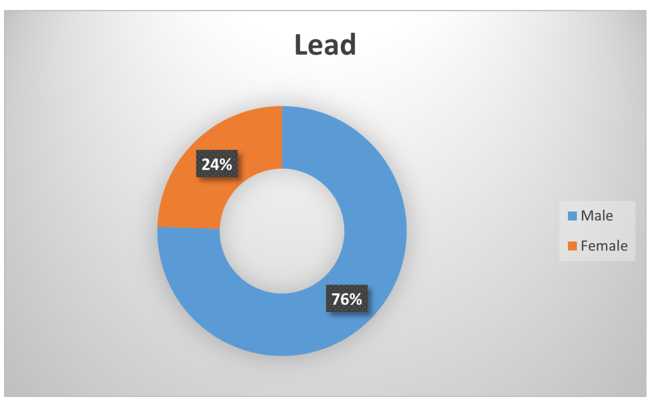

Figure 2: Male and Female Roles (Designed and Created by group using Matplotlib)

Q:2: Has gender balance in speaking roles changed over time (i.e., years)?
If we look at the data from 2000-2015, nothing much has changed in these 15 years. Numbers are still the same. Male speaking roles have been dominating (as shown in the graph below). In the year 2015, we can see some exceptions that the females were slightly dominant; to see how this goes further, we'll need some more data to analyze further.

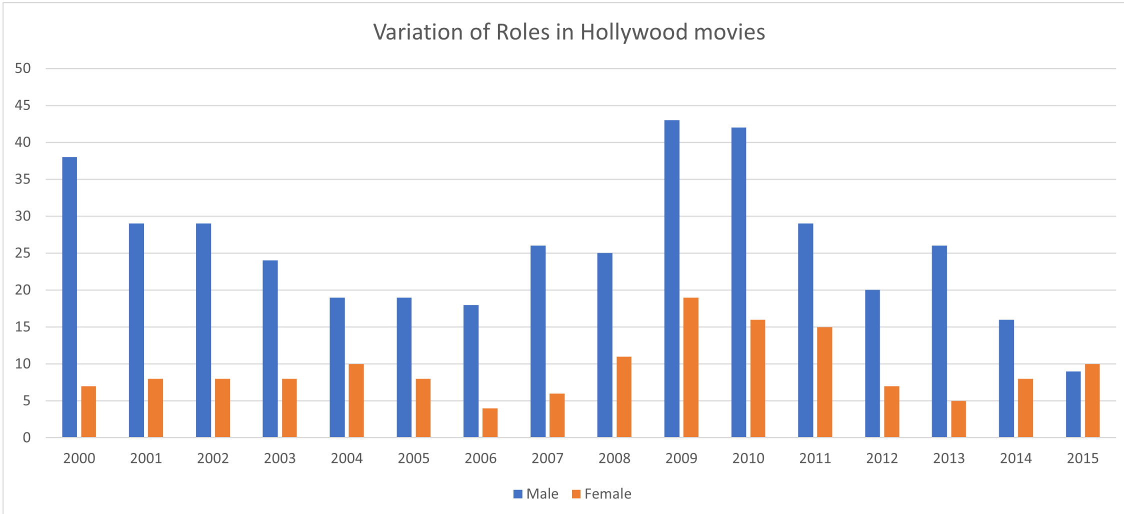

Figure 3: Male and Female gender balance (Designed and Created by group using Matplotlib)

Q:3: Do films in which men do more speaking make a lot more money than films in which women speak more?
Yes, the Below graph clearly shows that leading male films has done some really good business as compared to the leading female roles. Comparing the data from 1975 to 2015, we saw that there were clear and significant gaps between the revenue generated by leading male films and leading female films.

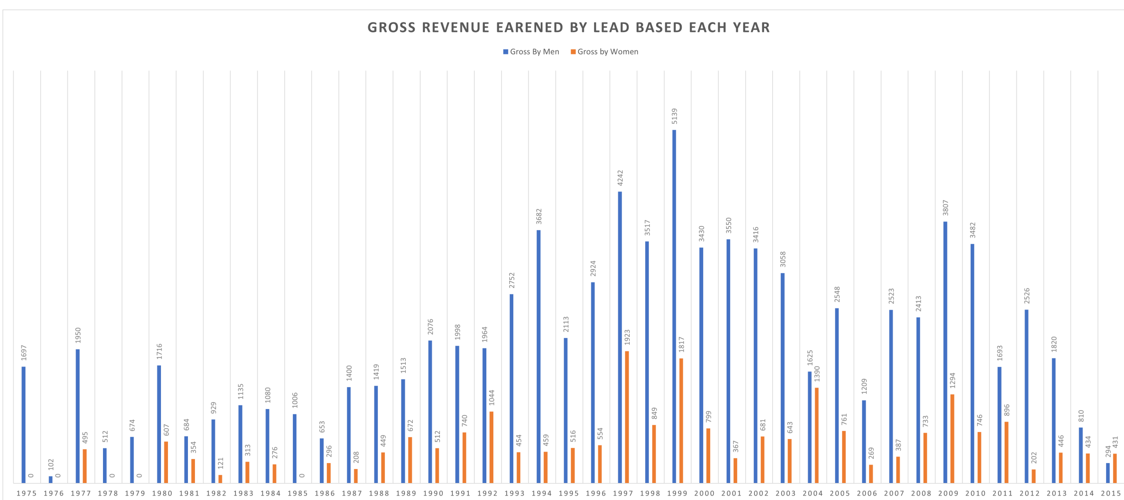

Figure 4: Role and Revenue (Designed and Created by group using Matplotlib)

3 Methodology

3.1 Logistic Regression

In Logistic Regression, we use the statistical analysis method on prior observation of the data set to get a binary outcome, like 1 or 0. The resultant prediction of logistic regression is a dependent data variable that we get after analyzing the relationship between one or more independent variables. [2]

Application and Evaluation:
As said in the definition, Logistic regression is used to predict the outcomes as true/false or 0/1, so in our case as well, we are here to predict the lead role in the films as Male(1)/Female(0). First, we

[2]https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression

imported the whole Train.csv file into the jupyter notebook, analyzed the dataset, saw how many male and female leads there were, and analyzed the dataset based on the age of leads. Further, since the Logistic Regression model predicts 0 and 1, we remade the Lead column in the form of 0(female) and 1(male). Based on the evaluation, we dropped some columns which might not be useful in making predictions. We took dependent and independent variables for our model from the dataset. After that, From the sklearn library, we applied the train test split module to our data set to stimulate how our model is going to work, having a test size of 33% and a random state of 42. After that we used, we used the LogisticRegression method from sklearn linear model on our dataset, through which we got the following results:
Train Accuracy = 85.9%
Test Accuracy = 84.0%
We also used the confusion metric to see how our model works. In order to tune our methods, we used hyperparameter regression with the following parameters penalty, C, solver, max iter, and using GridSearchCV module; we got the Accuracy = 87.9%.

3.2 Random Forest

Random Forest, which we use for classification or regression, is an supervised learning algorithm that implements an ensemble learning method that constitutes a large number of decision trees, the outputted result is the consensus of best answer to that problem. It is mostly used for classification and regression.
Ensemble learning, which is based on collective opinion, consists of many machine learning algorithms that combine to produce a better result- the wisdom of the crowd. Just like if we say many people with lesser knowledge on a topic can produce a better result by helping each other instead of one person with more knowledge on something performing all in all. [3]

Application and Evaluation:
As done for the logistic Regression, we used the process here as well till using the sklearn library for applying the train test split module to our data set with the test size of 33% and a random state of 5. After that we used, we used the ensemble.RandomForestClassifier method with number of estimators = 100 from sklearn, through which we got the following results:
Train Accuracy = 100%
Test Accuracy = 82.2%
As the model is overfitting, we tried to fix the model using the tuning method of hyper parameterization using the following parameters: estimators, max features, max depth, min samples split, min samples leaf, and bootstrap. After tunning the method with these parameters and then checking the accuracy of our model, we got the following results : Train Accuracy = 84.8%
Test Accuracy = 79.0%

3.3 Naive Classifier

Naive Bayes is actually a group of classification algorithms that uses Bayes' Theorem. The Principal mainly followed by that every pair of classified features is independent of each other.

Bayes' Theorem computes the probability for an event occurring given that the probability of another which already occurred. Naive Bayes' theorem is stated below in mathematical form as the following equation: Where A and B are events and P(B) $\neq 0$ $\quad P(A \mid B) = \frac{P(B|A)P(A)}{P(B)}$

P(A/B) posterior probability P(B/A) Likelihood P(A) Prior Probability P(B) Prior Probability that the Evidence is True

Multinomial Naive Bayes have a representation of frequencies using feature vectors which are used to produce certain events. This is the event model that is typically used for the purpose of document classification.
Bernoulli Naive Bayes have input features representation in the form of booleans variables that are independent of each other. Similar to the multinomial event model, this is also used for document

[3] https://www.nvidia.com/en-us/glossary/data-science/random-forest/

classification tasks, buts it's more popular, where binary means yes or no in words example, we say either words exist or not. We use features in there instead of term frequencies.
In Gaussian, Naive Bayes uses continuous values that are associated with features, and they are assumed to be distributed according to the Gaussian distribution. There is another name for this distribution Normal distribution. When we plot it, it shows a curve that appears bell-shaped, and it is symmetric about the mean of the feature values.

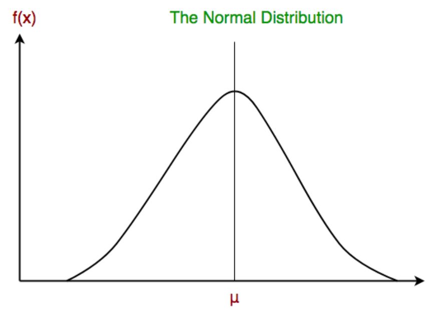

Figure 5: Gaussian Naive Bayes [4]

Application and Evaluation:
As done in the other algorithms, we used the process here as well, using the sklearn library for applying the train test split module to our data set with the test size of 33% and a random state of 5. After that we used, we used three classification algorithms from the Bayes theorem family of classification. We got the following results with algorithms.

Gaussian Naive Bayes:
Accuracy Score = 72%
Posterior Probability of POS Label 'Male'= 80%

Multinomial Naive Bayes:
Accuracy Score = 54%
Posterior Probability of POS Label 'Male' = 74%

Bernoulli Naive Bayes:
Accuracy Score = 77%
Posterior Probability of POS Label 'Male' = 87%

3.4 Method for Production

We will use it for production to decide on one model out of these two selected models; we carried out the cross-validation technique. The model which gives better accuracy with cross-validation will be the one to be taken in for production. So we carried out the K Fold Cross Validation technique for both models. By using the K Fold Cross Validation(KFold(10)) technique for Logistic Regression, we got 86% accuracy. But for Random forests, using the same technique, we got an accuracy of 85.36%. Based on these evaluations we got from the cross-validation technique, we can see that Logistic Regression is ahead, and we can go with this method for production.

References

https://www.nature.com/articles/s41599-020-0436-1/figures/1
https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression
https://www.nvidia.com/en-us/glossary/data-science/random-forest/
https://www.geeksforgeeks.org/naive-bayes-classifiers/

[4] https://www.geeksforgeeks.org/naive-bayes-classifiers/

4 Appendix

Explanation of data
Year: That the film was released.
Number of female actors : With major speaking roles.
Number of male actors : With major speaking roles.
Gross : Profits made by film.
Total words : Total number of words spoken in the film.
Number of words male : Number of words spoken by all other male actors in the film (excluding lead if lead is male)
Number of words female : Number of words spoken by all other female actors in the film (excluding lead if lead is female)
Number of words lead : Number of words spoken by lead.
Difference in words lead and co-lead : Difference in number of words by lead and the actor of opposite gender who speaks most.
Lead Age : Age of lead actor.
Co-lead Age : Age of co-lead actor.
Mean Age Male : Mean age of all male characters.
Mean Age Female : Mean age of all female characters.

4.1 Code for Logistic Regression

```

#step 1 : Import Modules
from sklearn.datasets import make_classification
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd
import numpy as np
import seaborn as sns


#step 2 : Import Dataset
df = pd.read_csv("train.csv")


#step 3 : Viewing the data
df.head()

#Data Analysis
#Visually Analysisng the data using Seaborn

#Check how many MALE and Female are there using countplot method

sns.countplot(x='Lead', data = df)

#null values in the dataset

df.isna().sum()


#Visualizing the null values
sns.heatmap(df.isna())

#data of age of leads where Lead is Male
df_Male = df.where(df['Lead'] == 'Male')
df_M=df_Male.dropna(how='all')
```

```

#data of age of leads where Lead is Female
df_Female = df.where(df[’Lead’] == ’Female’)
df_F=df_Female.dropna(how=’all’)

#find the distribution of age where Lead is Male
sns.displot(x=’Age Lead’, data= df_M)


#find the distribution of age where Lead is Female
sns.displot(x=’Age Lead’, data= df_F)


#Preparing data for model
#Convert Lead gender coloumn to binary values
pd.get_dummies(df[’Lead’])

Gender_B = pd.get_dummies(df[’Lead’], drop_first=True)
df[’Gender_B’] = Gender_B

df.head()

#drop column which are not required
df.drop([’Year’,’Gross’,’Age Co-Lead’],axis=1,inplace=True)

df.head()

#Seperate dependent and dependent variable
#independent variable
x = df[[’Total words’,’Number words male’,’Number words female’,’Number of words lead’,’Differenc
#dependent variable
y = df[’Gender_B’]
print(y)

#Data Modeling

from sklearn.model_selection import train_test_split

#train test split
x_train, x_test, y_train, y_test =  train_test_split(x,y,test_size=0.33, random_state=42)

import logistic regression
from sklearn.linear_model import LogisticRegression

#Fit Logistic Regression
lr = LogisticRegression()

lr.fit(x_train, y_train)

#check accuracy
print(f’Train Accuracy = {lr.score(x_train, y_train):.3f}’)
print(f’Test Accuracy = {lr.score(x_test, y_test):.3f}’)

#prediction
predict = lr.predict(x_test)

#Probabiltiy for +tive outcome is kept
lr_prob = lr.predict_proba(x_test)[:,1]

```

```
#Complete AREA UNDER THE ROC CURVE values
from sklearn.metrics import roc_curve, roc_auc_score
lr_auc = roc_auc_score(y_test, lr_prob)

#Display the AREA UNDER THE ROC CURVE score
print("Logistic Regression : AUROC = %.3f" %(lr_auc))

#Calculate AREA UNDER THE ROC CURVE score
lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_prob)

#plot the curve
plt.figure(figsize = (15,10))
plt.plot(lr_fpr, lr_tpr, linestyle = ’--’, label = ’Random forest (AUROC = %0.3f)’ %lr_auc)

plt.title(’ROC Plot’)
plt.xlabel(’False Positive Rate’)
plt.ylabel(’True Positive Rate’)

plt.legend()
plt.show()

#print Confusion Matrix to see how well your model works
from sklearn.metrics import confusion_matrix


pd.DataFrame(confusion_matrix(y_test,predict),  columns = [’Predicted No’, ’Predicted Yes’], inde


from sklearn.metrics import classification_report
print(classification_report(y_test, predict))


#Tunning the method using hyperparameter regression

logModel = LogisticRegression()

param_grid = [
    {’penalty’ : [’l1’,’l2’,’elasticnet’,’none’],
     ’C,’ : np.logspace(-4, 4, 20),
     ’solver’ : [’lbfgs’,’newton-cg’,’liblinear’,’sag’,’saga’],
     ’max_iter’ : [100, 1000, 2500, 5000]

    }
]


from sklearn.model_selection import GridSearchCV
clf = GridSearchCV(logModel, param_grid = param_grid , cv = 3, verbose = True, n_jobs = -1)

best_clf = clf.fit(x,y)
]:

best_clf.best_estimator_


#check accuracy
print(f’Accuracy = {best_clf.score(x,y):.3f}’)
```

```
from sklearn.model_selection import KFold
model = LogisticRegression()
kfold_validation = KFold(10)


from sklearn.model_selection import cross_val_score
result = cross_val_score(model, x, y, cv = kfold_validation)
print(result)
print(np.mean(result))

from sklearn.model_selection import ShuffleSplit
model = LogisticRegression()
ssplit = ShuffleSplit(n_splits = 10, test_size = 0.30)
results = cross_val_score(model, x, y, cv=ssplit)

results
print(f’Results - {results}’)
print(f’Mean Result - {np.mean(results):.5f}’)
```

4.2 Code for Random Forest Regression

```
#importing the modules
from sklearn.datasets import make_classification
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd
import numpy as np
import seaborn as sns

#reading the dataset
df = pd.read_csv("train.csv")
step 3 : Viewing the data
df.head()


##Data Analysis
##Visually Analysisng the data using Seaborn
##Check how many MALE and Female are there using countplot method

sns.countplot(x=’Lead’, data = df)
#Checking for null values in the dataset

df.isna().sum()

#Visualizing the null values in the dataset


sns.heatmap(df.isnull())

#data of age of leads where Lead is Male
df_Male = df.where(df[’Lead’] == ’Male’)
df_M=df_Male.dropna(how=’all’)

#data of age of leads where Lead is Female
df_Female = df.where(df[’Lead’] == ’Female’)
df_F=df_Female.dropna(how=’all’)
```

```
#find the distribution of age where Lead is Male
sns.displot(x='Age Lead', data= df_M)

#find the distribution of age where Lead is Female
sns.displot(x='Age Lead', data= df_F)


#Preparing data for model
#Convert Lead gender coloumn to Numerical values
pd.get_dummies(df['Lead'])


Gender_B = pd.get_dummies(df['Lead'], drop_first=True)


df['Gender_B'] = Gender_B


df.head()


#drop column which are not required for the model
df.drop(['Year','Gross','Age Co-Lead'],axis=1,inplace=True)

df.head()

#Seperate dependent and independent variable
#independent variable
x = df[['Total words','Number words male','Number words female','Number of words lead','Differenc
#dependent variable
y = df['Gender_B']

print(y)


#Data Modeling

from sklearn.model_selection import train_test_split

#train test split
x_train, x_test, y_train, y_test =  train_test_split(x,y,test_size=0.33, random_state=5)

import Random Forest
from sklearn import ensemble
rf_clf = ensemble.RandomForestClassifier(n_estimators = 100)
rf_clf.fit(x_train,y_train)


#check accuracy
print(f'Train Accuracy = {rf_clf.score(x_train, y_train):.3f}')
print(f'Test Accuracy = {rf_clf.score(x_test, y_test):.3f}')

#Predict Probabiltiy
rf_prob = rf_clf.predict_proba(x_test)

#Probabiltiy for +tive outcome is kept
rf_prob = rf_clf.predict_proba(x_test)[:,1]

#Complete AREA UNDER THE ROC CURVE values
```

```
from sklearn.metrics import roc_curve, roc_auc_score
rf_auc = roc_auc_score(y_test, rf_prob)

#Display the AREA UNDER THE ROC CURVE score
print("Random forest : AUROC = %.3f" %(rf_auc))

#Calculate AREA UNDER THE ROC CURVE score
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_prob)

#plot the curve
plt.figure(figsize = (15,10))
plt.plot(rf_fpr, rf_tpr, linestyle = ’--’, label = ’Random forest (AUROC = %0.3f)’ %rf_auc)

plt.title(’ROC Plot’)
plt.xlabel(’False Positive Rate’)
plt.ylabel(’True Positive Rate’)

plt.legend()
plt.show()

#No. of trees
n_estimators,  = [int(x) for x in np.linspace(start = 10, stop = 80, num = 10)]
#No. of features
max_features = [’auto’, ’sqrt’]
#Max number of levels in tree
max_depth = [2,4]
#Mini no. of samples to split node
min_samples_split = [2, 5]
#Mini no. of samples required at each leaf node
min_samples_leaf = [1, 2]
#Method of selecting samples for training each tree
bootstrap = [True, False]

#Create the param grid
param_grid = {’n_estimators’: n_estimators,
               ’max_features’: max_features,
               ’max_depth’: max_depth,
               ’min_samples_split’: min_samples_split,
               ’min_samples_leaf’: min_samples_leaf,
               ’bootstrap’: bootstrap}


rf_Model = ensemble.RandomForestClassifier()


from sklearn.model_selection import GridSearchCV
rf_Grid = GridSearchCV(estimator = rf_Model, param_grid = param_grid , cv = 3, verbose = 2, n_job


rf_Grid.fit(x_train,y_train)


rf_Grid.best_params_

#check accuracy
print(f’Train Accuracy = {rf_Grid.score(x_train, y_train):.3f}’)
print(f’Test Accuracy = {rf_Grid.score(x_test, y_test):.3f}’)

from sklearn.model_selection import KFold
```

```
model = ensemble.RandomForestClassifier()
kfold_validation = KFold(10)


from sklearn.model_selection import cross_val_score
result = cross_val_score(model, x, y, cv = kfold_validation)
print(result)
print(np.mean(result))



from sklearn.model_selection import ShuffleSplit
model = ensemble.RandomForestClassifier()
ssplit = ShuffleSplit(n_splits = 10, test_size = 0.30)
results = cross_val_score(model, x, y, cv=ssplit)

print(f'Results = {results}')
print(f'Mean Result = {np.mean(results):.5f}')
```

4.3 Code for Naive Classifier

```
#step 1 : Import Modules
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, f1_score

#step 2 : Import Dataset
df = pd.read_csv("train.csv")
# checking within the columns if we have missing values
df.info()

#step 3 : Viewing the data
df.head(5)

#step 4 : Separating the Labels for our continues other attributes
x = df.drop('Lead', axis=1)
y = df['Lead']

#step 5 : Splitting the training & test data
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=42)


# step 6 : Using Bernoulli Naive Bayes for score and also showing accuracy
BernNB = BernoulliNB(binarize= 0.1)
BernNB.fit(x_train, y_train)
print(BernNB)

y_expect = y_test
y_pred = BernNB.predict(x_test)
print("Accuracy Score: ",accuracy_score(y_expect, y_pred))
print("POS Label Male: ", f1_score(y_expect, y_pred, average="binary", pos_label="Male"))

# step 7 : Using Multinomial Naive Bayes for score and also showing accuracy
```

```
MultiNB = MultinomialNB()
MultiNB.fit(x_train, y_train)
print(MultiNB)
y_pred = MultiNB.predict(x_test)
print("Accuracy Score: ",accuracy_score(y_expect, y_pred))
print("POS Label Male: ", f1_score(y_expect, y_pred,average="binary", pos_label="Male"))

# step 8 : Using Gaussian Naive Bayes for score and also showing accuracy
GausNB = GaussianNB()
GausNB.fit(x_train, y_train)
print(GausNB)
y_pred = GausNB.predict(x_test)
print("Accuracy Score: ",accuracy_score(y_expect, y_pred))
print("POS Label Male: ", f1_score(y_expect, y_pred, average="binary", pos_label="Male"))
```