CE802-MACHINE LEARNING AND DATA MINING

PROFESSOR : Dr LUCA CITI

STUDENT NAME: DILPA RAO

STUDENT ID : 1906319

TABLE OF CONTENTS

INTRODUCTION :

The manager of a social media company is plagued with fake news and misleading information being posted on his company's platform and it causes damage to certain parties. We aim to solve this problem by using different machine learning algorithm which are described below to identify the source.

1.*Pilot – Study Proposal* (661- words)

Project Details :

A. TYPES OF PREDICTIVE TASKS THAT NEEDS TO BE PERFORMED

This is a **supervised learning binary classification problem** as we need to train the model and predict whether the news is fake or not.
There are many classification algorithm that we can use like (Decision Tree, K-NN, SVM, Random Forest, Logistic Regression, XGboost, CART). We will be using the first four of the above mentioned classifiers. The various reasons for choosing these classifiers for fake news detection are as follows:

1. **Decision Tree Classifier:**

   - Powerful tool for classification
   - Can be easily visualised and easy to explain
   - Does not require normalization and scaling of data
   - Requires less effort for data preparation during pre-processing as compared to other classifiers.
   - Handles both categorical and numerical features
   - Provide a clear indication as to which feature are the important for the classification or prediction

2. **K Nearest Neighbor Classifier (K-NN)**
   - Faster than other algorithms that require training
   - Requires no training before making prediction, new data can be added which will not impact the accuracy of the algorithm
   - Easy to implement as it required only two parameters – the value of k, the distance function (Manhattan or Euclidean).

- Easily Interpreted

3. Support Vector Machine Classifier (SVM)

- Offer good accuracy, use less memory and perform faster prediction as compared to some other algorithms like the Naïve Bayes.
- Good generalization capabilities which prevents it from over-fitting
- Stable, a small change to the data does not greatly affect the hyperplane.
- Works well when we have no idea on the data, can work on structured and semi-structured data like texts, images etc.

4. Random Forest Classifier

- Stable algorithm ,as compared to other algorithm it is less impacted by noise
- Standardisation and Normalisation is not required as it uses rule based approach instead of distance approach
- Robust to outliers and can handle them automatically
- Based on bagging algorithm and uses ensemble learning technique
- Solves the problem of overfitting in decision tree by creating many subsets of trees and combining the output of all the trees

B. POSSIBLE INFORMATIVE FEATURES NEEDED

Table 1. List of features

No	Feature	Description
1.	Source	Origin of the post was it newspaper, articles, tv stations or individual user
2.	User Id	Id of the user of this account
3.	Frequency of Posts	How many times does the user posts
4.	Time of Posts	At what time does posting normally happens early morning, day, evening, night
5.	Content of Post	The actual content of the post and whether its text, image or video
6.	Type of Post	Whether it is a long post or a short one
7.	Number of likes	How many likes were reported

8.	Date of post	What date the information was posted
9.	Number of connection	Number of connections the user have on the platform
10.	Comments	Type of comments on the post
11.	Country	Which country was the account created
12.	Label	Whether it is Fake or Real

C. EVALUATING THE PERFORMANCE BEFORE DEPLOYMENT

We have used the following metrics to evaluate the performance of the machine learning classifier algorithm.

1. ACCURACY

It is the ratio of the correctly classified labels to the total number of samples in the dataset. Higher ratio denotes greater accuracy.

2. KAPPA STATISTICS

It measures how good a classifier is by comparing its performance with chance guesses of the label. It is more robust than a simple agreement calculation as k considers the possibility of the chance agreement .

A kappa statistic of1 implies a perfect predictor whereas 0 means that the classifier provides no information, it behaves as if it is randomly guessing.

2. *COMPARATIVE STUDY* (1221- words)

Analysing the dataset :

We were provided with the datasets 'CE802_Ass_2019_Data.csv' and CE802_Ass_2019_Test.csv' , the former was to be used as a training and after finding the proper classifier we were supposed to use that classifier and populate the values in the test set.

Given below in the table is the statistical distribution of the dataset.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
count	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	328.0
mean	0.7	0.5	93.2	-0.1	-0.0	0.0	1.2	-0.0	38.4	7.2	-0.0	-0.0	0.9	0.7	0.0	0.4	-0.1	-0.1	9.3	0.1
std	0.5	0.5	70.3	1.5	1.0	1.0	1.4	1.0	25.3	16.7	1.1	1.0	0.7	15.5	1.0	1.6	1.0	1.0	5.8	1.9
min	0.0	0.0	3.0	-4.3	-2.9	-2.9	-2.3	-3.2	0.0	-49.0	-3.5	-3.6	0.0	-41.0	-3.2	-4.1	-3.1	-3.2	0.5	-5.4
25%	0.0	0.0	37.0	-1.1	-0.8	-0.7	0.3	-0.7	18.0	-5.0	-0.8	-0.7	0.3	-9.2	-0.6	-0.8	-0.7	-0.7	5.1	-0.7
50%	1.0	1.0	79.0	-0.1	-0.0	0.0	1.2	-0.1	35.0	7.0	0.0	0.0	0.8	0.0	-0.0	0.4	-0.1	-0.1	8.2	0.2
75%	1.0	1.0	135.2	1.0	0.6	0.8	2.1	0.6	54.0	19.0	0.8	0.7	1.3	12.0	0.7	1.5	0.6	0.6	11.8	0.9
max	1.0	1.0	338.0	3.9	3.3	3.2	5.7	2.7	140.0	61.0	3.9	3.0	4.0	55.0	3.3	5.7	2.6	3.6	36.8	6.1

On analysing the training dataset which consisted of 20 features ,500 samples, and the column 21 which has the label value of True / False for the fake news, it was found that in column F20 there were 172 missing values . All the other columns didn't have any missing values.

```
F1          0
F2          0
F3          0
F4          0
F5          0
F6          0
F7          0
F8          0
F9          0
F10         0
F11         0
F12         0
F13         0
F14         0
F15         0
F16         0
F17         0
F18         0
F19         0
F20       172
Class       0
dtype: int64
```

We could have altogether excluded this variable ,but since the missing values comprised only 34 % , we decided to impute it instead of leaving it as it can greatly effect the outcome of the

predictor Class. We could have done either of mean, median or mode imputation. The mode value was -0.13 and the median was 0.17.Since there was not a lot of variation, we decided to use the mean value for the imputation which was 0.074.

GridSearchCV :

We have used GridSearchCV method to search the best parameters of the classifier from the grid of given parameters. It is useful for finding the best parameter for the target model and dataset. In this method multiple parameters are tested by cross validation method and the best parameters are extracted to apply for a predictive model.

We have passed the grid of parameters in the GridSearchCV and using k-fold=5 on all the four classifiers and obtained the best parameters.

Comparison of different Classifier Modelling Algorithm :

All the four classifiers used the stratified 5-fold cross validation technique. Given below is an explanation of how the 5-fold technique works.

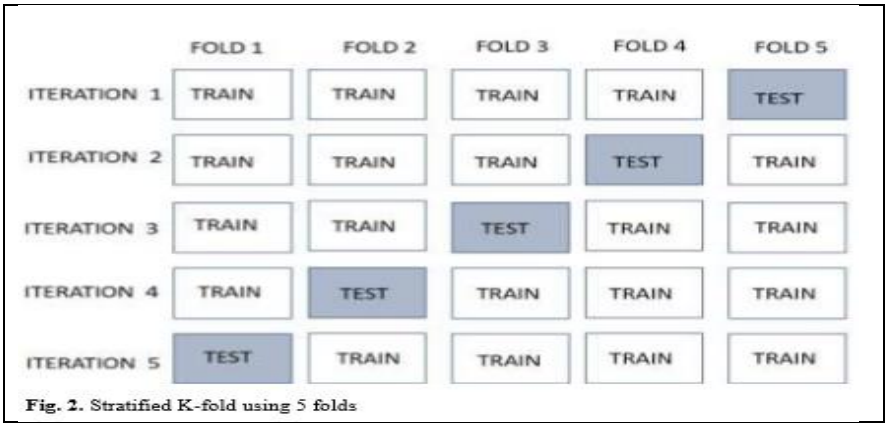


Fig. 2. Stratified K-fold using 5 folds

In this the data is divided into 5 folds , 4 folds constitute the training set and 1 fold constitute the test set. Each fold in turn becomes a test set whereas the remaining four are used as training. The performance is then averaged. Given below in the table is the 5-fold cross validation score for all the four classifiers.

Classifier	score
Random Forest	0.600148
Decision Tree	0.597410
Support Vector Machines(Kernel)	0.592112
K Nearest Neighbor	0.570919

From the above table it clearly shows the accuracy rate of the different classifiers and it is observed that the Random Forest Classifier performs the best at 65%, followed by K Nearest Neighbor 62%, Decision Tree 60% and finally Support Vector Machines 60 %.

1. **Decision Tree Classifier Algorithm :**

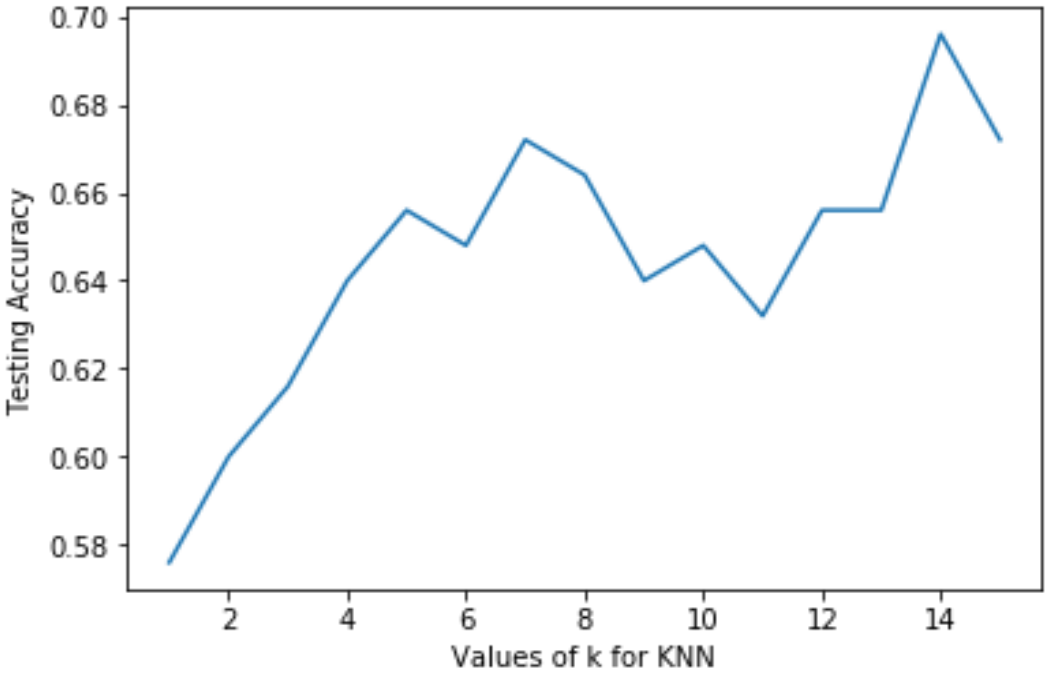
   The best parameters for decision tree was as follows :
   (the best results are :{'class_weight' : None, 'criterion' : 'entropy', 'max depth' : 3, 'presort' : True}).
   On passing these best parameters to our training set and predicting the test set an **accuracy score of 56.8 %** was achieved by the Decision Tree Classifier. Furthermore, on doing Kappa Statistics we got a score of 0.028 which is not so impressive. Moreover, these are some of the following limitations.
   - Small change in the data can cause a large change in the structure of the tree
   - Requires more time to train the model
   - More complex calculations compared to other algorithms

2. **Random Forest Classifier :**

   A Random Forest Classifier fits a number of decision tree classifiers on various randomly selected samples and it averages out to improve accuracy and prevent over-fitting.
   The best parameters for the Random Forest Classifier was as follows :
   (the best results are : { 'criterion' : 'gini', 'max_features' : 'auto', 'min_samples_leaf' : 2, 'min_samples_split' : 6, 'n_estimators' : 100, 'n_jobs' : -1, 'oob_score' : True, 'random_state' : 160, 'warm_start' : True }).
   An **accuracy score of 57.6 %** was achieved by the Random Forest Classifier which is same as that of the Decision Tree. Moreover, a Kappa statistic of 0.11 was observed. The points mentioned below need to be taken into consideration.
   - Classification made by Random forest is difficult to interpret by humans
   - Biased towards attributes with more levels
   - Smaller groups are preferred over larger groups ,when data contains correlated features
   - Overfits data with noisy classification

3. **K Nearest Neighbor :**

   In K Nearest Neighbor we implemented a loop to check the values of K from 1 to 15 and see out of the 15 values of K which gives the maximum accuracy.



The accuracy scores of k from 1 to 15 is as follows
[0.576, 0.6, 0.616, 0.64, 0.656, 0.648, 0.672, 0.664, 0.64, 0.648, 0.632, 0.656, 0.656, 0.696, 0.672]
From the graph below it clearly indicates the **maximum accuracy score** is when K=14 which is **69.6 %.**
A Kappa statistic of 0.36 was observed.
So, we made the K Nearest Neighbor Classifier using K=14.
However, in spite of good accuracy we should take into account the features mentioned below and look for a better and optimised classifier.

- Easy and simple but as the dataset grows the efficiency and speed of the algorithm detoriates
- It is very sensitive to outliers
- Does not perform well on imbalance data
- Cannot deal with missing numbers
- Biggest issue is choosing the optimal number of neighbors

4. **Support Vector Machines with kernel :**

Support Vector Machine can give the highest accuracy rate after the training phase. We can use the soft margin or the hard margin approach based on our requirements. We have again used the GridSearchCV method to find the best parameter for our dataset which is as follows :
(the best results are : { 'C' : 10, 'gamma'' : 'auto', 'kernel' : 'linear' }).
It gives an **accuracy score of 68.8%** and a kappa statistic of 0.34.

There are some limitations to SVM , when it comes to choosing the appropriate kernel function and feature scaling of variables but, in spite of this it has good generalization and prevents over-fitting and it is more stable as small changes in the data does not affect the hyperplane.

So, based on the comparison of all four of our classifiers we will be using the Support Vector Classifier because of its high accuracy score and moreover, it only depends on a subset of points the support vector, so it is able to integrate the rest of the dataset more efficiently and quickly as compared to the other machine learning algorithm.

CONCLUSION :

In our report, we have evaluated the different machine learning algorithm to predict the accounts which are posting fake news or misleading information . We found that Support Vector Machine Classifier and Random Forest Classifier exhibit the highest accuracy score.

Machine Learning Algorithm	**Kappa Score**	**Accuracy Score**
Pruned Decision Tree Classifier	0.028	56.8 %
K Nearest Neighbor	0.36	69.6%
Random Forest	0.11	57.6%
Support Vector Machine with Kernel	0.34	68.8 %

We have trained the model by splitting the data into training and test size of 75% and 25% and by using random state 160. On varying the training and test datasets ratio (80-20, 85-15) and changing the random state(6, 42,101) we obtained similar results . Of all the classifier SVM and K-NN classifier were the ones with the highest accuracy score. Based on the comparative drawbacks of the other classifiers as mentioned above , it is prudent to predict the class using the SVM classifier

3. ***PREDICTION ON TEST SET***

We will be using the Support Vector Machine Classifier algorithm for predicting the Class of the ‘CE802_Ass_2019_test.csv ‘ file.

References :

Http://www.datacamp.com

http://www.kaggle.com

http://www.datatechnotes.com

https://scikit-learn.org

https://www.fromthegenesis.com

APPENDIX

SAMPLE CODE : JUPYTER NOTEBOOK CODE DOCUMENTATION

In [ ]:
```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```
# IMPORTING THE DATASET FILE ('DATA.CSV') USING PANDAS AND DISPLAYING THE FIRST FIVE ROWS USING HEAD().
```

In [ ]:
```
datadf=pd.read_csv('CE802_Ass_2019_Data.csv')
datadf.head()
```

In [ ]:
```
#CALCULATING THE TOTAL NA VALUES AND IT ALL BELONGS TO THE F20 COLUMN
```

In [3]:
```
import pandas_profiling
datadf.isna().sum()
```

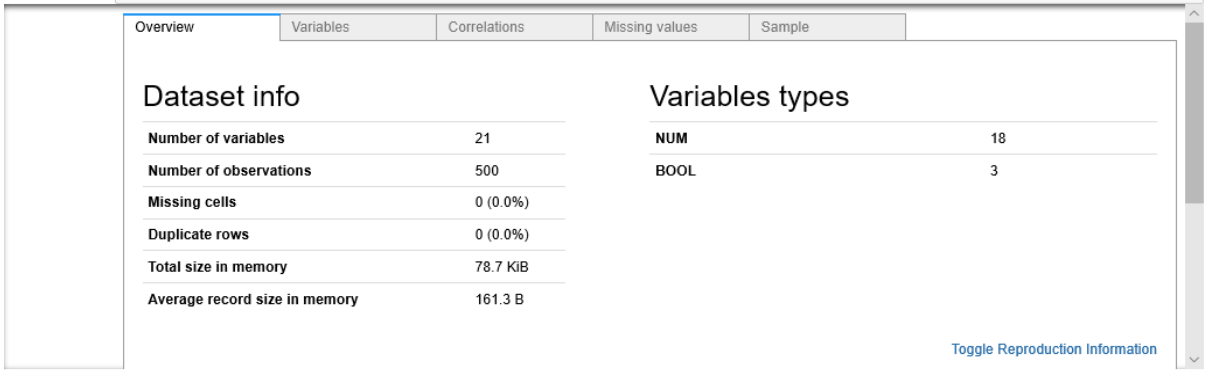
Out[3]:
```
F1         0
F2         0
F3         0
F4         0
F5         0
F6         0
F7         0
F8         0
F9         0
F10        0
F11        0
F12        0
F13        0
F14        0
F15        0
F16        0
F17        0
F18        0
F19        0
F20      172
Class      0
```

In [37]:
```
! pip install pandas-profiling
```

In [38]:
```
import pandas_profiling as pdp
```

In [39]:
```
report = pdp.ProfileReport(datadf,title='Pandas Profiling Report')
```

In [40]:
```
report
```



In [ ]:
```
# DOING THE IMPUTATION OF THE F20 COLUMN USING MEDIAN
```

In [7]:
```
datadf=pd.read_csv('CE802_Ass_2019_Data.csv')
datadf=round(datadf.fillna(datadf['F20'].median()),2)
datadf.head()
```

Out[7]:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	...	F12	F13	F14	F15	F16	F17	F18	F19	F20	Class
0	0	0	16	2.02	0.52	-2.35	-1.98	-0.70	85	6	...	-0.07	1.08	15	-0.63	-3.49	-1.68	0.02	15.3	0.17	True
1	0	0	86	-0.90	2.75	0.14	0.83	-0.06	107	1	...	0.17	1.06	-8	-1.21	0.34	0.36	0.61	10.1	0.17	True
2	1	1	165	0.73	1.05	0.10	2.57	-1.65	41	5	...	0.04	0.42	-6	-0.46	-0.62	1.67	2.60	11.0	1.55	False
3	1	1	191	-1.50	0.79	0.33	1.24	1.35	17	2	...	1.74	1.74	15	0.47	0.63	0.08	0.19	6.3	0.95	False
4	1	1	13	0.25	-1.19	-0.90	2.67	0.22	12	8	...	-0.39	1.25	25	-0.09	-2.41	-0.53	-0.77	10.5	0.17	True

5 rows × 21 columns

In [ ]:
```
# CALCULATING THE MODE OF F20 COLUMN AND SINCE ITS A NEGATIVE VALUE NOT DOING ITS IMPUTATION
```

In [8]:
```
datadf=pd.read_csv('CE802_Ass_2019_Data.csv')
datadf=datadf.fillna(datadf['F20'].mean())
datadf.head()
```

Out[8]:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	...	F12	F13	F14	F15	F16	F17	F18	F19	F20	Class
0	0	0	16	2.02	0.52	-2.35	-1.98	-0.70	85	6	...	-0.07	1.08	15	-0.63	-3.49	-1.68	0.02	15.3	0.074146	True
1	0	0	86	-0.90	2.75	0.14	0.83	-0.06	107	1	...	0.17	1.06	-8	-1.21	0.34	0.36	0.61	10.1	0.074146	True
2	1	1	165	0.73	1.05	0.10	2.57	-1.65	41	5	...	0.04	0.42	-6	-0.46	-0.62	1.67	2.60	11.0	1.550000	False
3	1	1	191	-1.50	0.79	0.33	1.24	1.35	17	2	...	1.74	1.74	15	0.47	0.63	0.08	0.19	6.3	0.950000	False
4	1	1	13	0.25	-1.19	-0.90	2.67	0.22	12	8	...	-0.39	1.25	25	-0.09	-2.41	-0.53	-0.77	10.5	0.074146	True

5 rows × 21 columns

In [9]:
```
datadf=pd.read_csv('CE802_Ass_2019_Data.csv')
datadf['F20'].mode()
```

Out[9]:
```
0   -0.13
dtype: float64
```

In [ ]:
```
# FINALLY DOING THE IMPUTATION OF THE F20 COLUMN USING THE MEAN() AND USING THIS AS THE FINAL DATASET TO DO PREDICTIONS.
```

In [ ]:
```
# SEPARATING THE TARGETS AND THE LABELS INTO X,Y. ALL THE COLUMNS TILL F20 GOES IN X AND ONLY THE CLASS COLUMN COLUMN IS IN Y.
```

In [ ]:
```
datadf=pd.read_csv('CE802_Ass_2019_Data.csv')
datadf=datadf.fillna(datadf['F20'].mean())
x=datadf.iloc[:,datadf.columns!='Class'].values
y= datadf.iloc[:,20].values
```

In [ ]:
```
# IMPORTING THE DIFFERENT LIBRARIES FROM SKLEARN FOR BUILDING THE CLASSIFIER MODEL AND USING THE TEST_TRAIN_SPLIT FUNCTION TO
# DIVIDE THE DATA INTO TRAINING SET AND TESTING SET FOR 75% AND 25% AND USING THE STRATIFY .
```

In [ ]:
```
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import sklearn.model_selection as model_selection
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn import metrics
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25,random_state=160,stratify=y)
```

In [ ]:
```
# USING THE STANDARD SCALER FUNCTION TO SCALE THE DATA BUT SINCE THE DATA IS NOT SO WIDELY DISTRIBUTED NOT USING IT .
```

In [ ]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [ ]:
```
# CALCULATING THE BEST PARAMETERS FOR DECISION TREE USING THE GRID SEARCH AND DOING 5 FOLD CROSS VALIDATION
```

In [20]:
```
parameters_dt = {'criterion':('gini', 'entropy'),
                 'max_depth':[3,4,5,6],
                 'class_weight':('balanced', None),
                 'presort':(False,True),
                }
dt = DecisionTreeClassifier()
clf_dt = GridSearchCV(dt, parameters_dt,cv = 5, scoring = 'accuracy')
clf_dt1=clf_dt.fit(X_train, y_train)

def display(results):
    print(f'the best results are :{results.best_params_}')
display(clf_dt)


dt_pred=clf_dt1.predict(X_test)
accuracy_score(y_test,dt_pred)
```

```
the best results are :{'class_weight': None, 'criterion': 'entropy', 'max_depth': 3, 'presort': True}
```

```
C:\Users\dilpa\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:814: DeprecationWarning: The default of the `iid`
parameter will change from True to False in version 0.22 and will be removed in 0.24. This will change numeric results when tes
t-set sizes are unequal.
  DeprecationWarning)
```

Out[20]:
```
0.568
```

In [ ]:
```
# USING THE BEST PARAMETERS :{'class_weight': None, 'criterion': 'entropy', 'max_depth': 3, 'presort': True}
#AND MAKING A MODEL WITH 56.8% ACCURACY
```

In [29]:
```
tree = DecisionTreeClassifier(max_depth=3,criterion='entropy',presort= True,random_state=160)
clf_dt=tree.fit(X_train, y_train)

dt_pred = clf_dt.predict(X_test)
dt_pred
accuracy_dt=accuracy_score(y_test, dt_pred)*100
print("Accuracy with Pruned Decision Tree Classifier :"+ str(accuracy_dt)+" %")
metrics.cohen_kappa_score(dt_pred,y_test)
```

```
Accuracy with Pruned Decision Tree Classifier :56.8 %
```

Out[29]:
```
0.02877697841726612
```

In [30]:
```
# LOADING THE CE802_ASS_2019_TEST DATA AND IMPUTING ITS F20 COLUMN WITH THE MEAN AND THEN USING THE DECISION TREE MODEL TO FILL
#COLUMN CLASS WITH THE VALUES OF FAKE NEWS TRUE AND FALSE.
```

In [36]:
```
test_data=pd.read_csv('CE802_Ass_2019_Test.csv')

f20_mean=test_data['F20'].mean()
test_data['F20'].fillna(f20_mean,inplace=True)

X_final=test_data.iloc[:,0:20].values
y_final=test_data.iloc[:,20].values


dt_y_final=clf_dt.predict(X_final)
dt_y_final

test_data['Class']=dt_y_final
test_data.to_csv("dt_test.csv")
```

In [ ]:
```
# CALCULATING THE BEST PARAMETER FOR THE SUPPORT VECTOR MACHINE WITH KERNEL USING THE GRIDSEARCHCV AND USING THE MODEL
```

In [37]:
```
svm = SVC()

parameters_svm =parameters = {
    "kernel": ["linear","rbf"], 'gamma':['auto'],
    "C":[0.1,1,10]
}


clf_svm=GridSearchCV(svm, parameters_svm,cv = 5, scoring = 'accuracy')
clf_svm1=clf_svm.fit(X_train, y_train)


def display(results):
    print(f'the best results are :{results.best_params_}')
display(clf_svm)

svm_pred=clf_svm1.predict(X_test)
accuracy_score(y_test,svm_pred)
```

```
C:\Users\dilpa\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:814: DeprecationWarning: The default of the `iid`
parameter will change from True to False in version 0.22 and will be removed in 0.24. This will change numeric results when tes
t-set sizes are unequal.
  DeprecationWarning)
```

```
the best results are :{'C': 10, 'gamma': 'auto', 'kernel': 'linear'}
```

Out[37]:
```
0.688
```

In [ ]:
```
# USING THE SVM WITH :{'C': 10, 'gamma': 'auto', 'kernel': 'linear'} PARAMETERS GIVE AN ACCURACY OF 68.8%
```

In [38]:
```
svc=SVC(gamma='auto',kernel='linear',C=10)
svm=svc.fit(X_train,y_train)
svm_pred=svm.predict(X_test)


accuracy_svm=accuracy_score(y_test,svm_pred)*100
print("Accuracy with Support Vector Machine with Kernel :"+ str(accuracy_svm)+" %")
metrics.cohen_kappa_score(svm_pred,y_test)
```

```
Accuracy with Support Vector Machine with Kernel :68.8 %
```

Out[38]:
```
0.3398781313473257
```

In [ ]:
```
# LOADING THE CE802_ASS_2019_TEST DATA AND IMPUTING ITS F20 COLUMN WITH THE MEAN AND THEN USING THE SUPPORT VECTOR KERNEL MODEL
#TO FILL THE COLUMN CLASS WITH THE VALUES OF FAKE NEWS TRUE AND FALSE USING THE svm MODEL.
```

In [39]:
```
test_data=pd.read_csv('CE802_Ass_2019_Test.csv')
f20_mean=test_data['F20'].mean()
test_data['F20'].fillna(f20_mean,inplace=True)

X_final=test_data.iloc[:,0:20].values
y_final=test_data.iloc[:,20].values


svm_y_final=svm.predict(X_final)
svm_y_final


test_data['Class']=svm_y_final
test_data.to_csv("svm_test.csv")
```

In [ ]:
```
# CALCULATING THE BEST PARAMETER FOR THE RANDOM FOREST CLASSIFIER USING THE GRIDSEARCHCV AND USING THE MODEL
```

In [40]:
```
rf = RandomForestClassifier()

parameters_rf = {'n_estimators': [100,50], 'max_features': ['auto'],
                 'n_jobs': [-1], 'min_samples_leaf': [2,4,], 'random_state':[160],
                 'min_samples_split':[2,6,], 'oob_score': [True,False],
                 'criterion': ['gini'], 'warm_start': [True,False]}


clf_rf=GridSearchCV(rf, parameters_rf,cv = 5, scoring = 'accuracy')
clf_rf1=clf_rf.fit(X_train, y_train)


def display(results):
    print(f'the best results are :{results.best_params_}')
display(clf_rf)


rf_pred=clf_rf1.predict(X_test)
accuracy_score(y_test,rf_pred)
```

```
the best results are :{'criterion': 'gini', 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 6, 'n_estimator
s': 100, 'n_jobs': -1, 'oob_score': True, 'random_state': 160, 'warm_start': True}
```

Out[40]:
```
0.576
```

In [ ]:
```
# USING THE RANDOM FOREST CLASSIFIER WITH:{'criterion': 'gini', 'max_features': 'auto', 'min_samples_leaf': 2,
#'min_samples_split': 6, 'n_estimators': 100, 'n_jobs': -1, 'oob_score': True, 'random_state': 160, 'warm_start': True}
#PARAMETERS GIVE AN ACCURACY OF 57.6%
```

In [41]:
```
rf=RandomForestClassifier(criterion= "gini",max_features="auto", min_samples_leaf= 2, min_samples_split=6, n_estimators=100, n_j
rf1=rf.fit(X_train,y_train)
rf_pred=rf1.predict(X_test)

accuracy_rf=accuracy_score(y_test,rf_pred)*100
print("Accuracy with Random Forest Classifier :"+ str(accuracy_rf)+" %")
metrics.cohen_kappa_score(rf_pred,y_test)
```

```
Accuracy with Random Forest Classifier :57.599999999999994 %
```

Out[41]:
```
0.11014103425117527
```

In [ ]:
```
# LOADING THE CE802_ASS_2019_TEST DATA AND IMPUTING ITS F20 COLUMN WITH THE MEAN AND THEN USING THE RANDOM FOREST CLASSIFIER
# MODEL TO FILL COLUMN CLASS WITH THE VALUES OF FAKE NEWS TRUE AND FALSE AND USING THE RF1 MODEL.
```

In [42]:
```
test_data=pd.read_csv('CE802_Ass_2019_Test.csv')
f20_mean=test_data['F20'].mean()
test_data['F20'].fillna(f20_mean,inplace=True)

X_final=test_data.iloc[:,0:20].values
y_final=test_data.iloc[:,20].values


rf_y_final=rf1.predict(X_final)
rf_y_final


test_data['Class']=rf_y_final
test_data.to_csv("rf_test.csv")
```

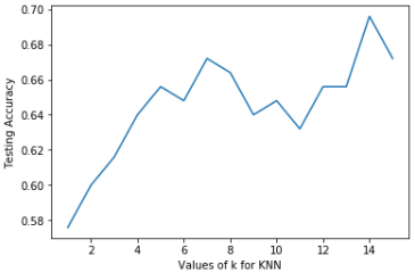
In [ ]:
```
# USING THE K NEAREST NEIGHBOR CLASSIFIER AND FINDING THE BEST VALUE OF K FROM 1 TO 15
```

In [43]:
```
# appending the values of the testing accuracy of the model prediction knn  for the values of k from 1 to 16 in the scores list.
k_range=range(1,16)
scores=[]
for k in k_range:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,y_train)
    knn_pred=knn.predict(X_test)
    scores.append(metrics.accuracy_score(y_test,knn_pred))
print(scores)
```

```
[0.576, 0.6, 0.616, 0.64, 0.656, 0.648, 0.672, 0.664, 0.64, 0.648, 0.632, 0.656, 0.656, 0.696, 0.672]
```

In [44]:
```
# The graph clearly shows the highest value of accuracy is 59% when k=1
plt.plot(k_range,scores)
plt.xlabel('Values of k for KNN')
plt.ylabel('Testing Accuracy')
```

Out[44]:
```
Text(0, 0.5, 'Testing Accuracy')
```



In [ ]:
```
# THE NEAREST NEIGHBOR IS 14 AND SCORE IS 69.6%
```

In [ ]:
```
# USING THE K NEAREST 14 NEIGHBOR ON THE TESTING SET AND GETTING THE ACCURACY 69.6%
```

In [45]:
```
clf_knn= KNeighborsClassifier(14)
clf_knn=clf_knn.fit(X_train,y_train)
knn_pred=clf_knn.predict(X_test)
accuracy_knn=round(accuracy_score(y_test,knn_pred)*100,2)
print("Accuracy with K :"+ str(accuracy_knn)+" %")

metrics.cohen_kappa_score(knn_pred,y_test)
```

```
Accuracy with K :69.6 %
```

Out[45]:
```
0.3606998654104979
```

In [ ]:
```
# LOADING THE CE802_ASS_2019_TEST DATA AND IMPUTING ITS F20 COLUMN WITH THE MEAN AND THEN USING THE K NEAREST NEIGHBOR
# CLASSIFIER MODEL TO FILL COLUMN CLASS WITH THE VALUES OF FAKE NEWS TRUE AND FALSE AND USING THE clf_knn MODEL.
```

In [47]:
```
test_data=pd.read_csv('CE802_Ass_2019_Test.csv')
f20_mean=test_data['F20'].mean()
test_data['F20'].fillna(f20_mean,inplace=True)

X_final=test_data.iloc[:,0:20].values
y_final=test_data.iloc[:,20].values


knn_y_final=clf_knn.predict(X_final)
knn_y_final


test_data['Class']=knn_y_final
test_data.to_csv("knn_test.csv")
```

In [ ]:
```
# COMPARING THE DIFFERENT CLASSIFIER ALGORITHM USING KFOLD 5 TIMES.
```

In [36]:
```

Classifiers=('Decision Tree','Support Vector Machines(Kernel)','Random Forest','K Nearest Neighbor')
scores = []
models = [clf_dt,svm,rf1,clf_knn]
for model in models:
    score = cross_val_score(model, X_train, y_train, scoring = 'accuracy', cv = 5, n_jobs = -1).mean()
    scores.append(score)
```

In [ ]:
```
mode = pd.DataFrame(scores, index = Classifiers, columns = ['score']).sort_values(by = 'score',
             ascending = False)
```

In [34]:
```
mode
```

Out[34]:

	score
Random Forest	0.600148
Decision Tree	0.597410
Support Vector Machines(Kernel)	0.592112
K Nearest Neighbor	0.570919

In [ ]:
```

```

CODE SNIPPETS

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 14 19:28:27 2020
@author: Dilpa Rao 1906319
"""
#Importing the libraries
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import pandas_profiling
%matplotlib inline
# IMPORTING THE DATASET FILE ('DATA.CSV') USING PANDAS AND DISPLAYING THE FIRST FIVE ROWS USING HEAD().datadf=pd.read_csv('CE802_Ass_2019_Data.csv')
datadf=pd.read_csv('CE802_Ass_2019_Data.csv')
datadf.head()

#CALCULATING THE TOTAL NA VALUES AND IT ALL BELONGS TO THE F20 COLUMN
datadf.isna().sum()
! pip install pandas-profiling
import pandas_profiling as pdp
report = pdp.ProfileReport(datadf,title='Pandas Profiling Report')
report

# DOING THE IMPUTATION OF THE F20 COLUMN USING MEDIAN
datadf=pd.read_csv('CE802_Ass_2019_Data.csv')
```

```
datadf=round(datadf.fillna(datadf['F20'].median()),2)

datadf.head()


# CALCULATING THE MODE OF F20 COLUMN AND SINCE ITS A NEGATIVE VALUE NOT DOING ITS IMPUTATION

datadf=pd.read_csv('CE802_Ass_2019_Data.csv')

datadf['F20'].mode()


# FINALLY DOING THE IMPUTATION OF THE F20 COLUMN USING THE MEAN() AND USING THIS AS THE FINAL DATASET TO DO PREDICTIONS.

# SEPARATING THE TARGETS AND THE LABELS INTO X,Y. ALL THE COLUMNS TILL F20 GOES IN X AND ONLY THE CLASS COLUMN IS IN Y.

datadf=pd.read_csv('CE802_Ass_2019_Data.csv')

datadf=datadf.fillna(datadf['F20'].mean())

x=datadf.iloc[:,datadf.columns!='Class'].values

y= datadf.iloc[:,20].values


# IMPORTING THE DIFFERENT LIBRARIES FROM SKLEARN FOR BUILDING THE CLASSIFIER MODEL AND USING THE TEST_TRAIN_SPLIT FUNCTION TO

# DIVIDE THE DATA INTO TRAINING SET AND TESTING SET FOR 75% AND 25% AND USING THE STRATIFY .

from sklearn.model_selection import train_test_split, StratifiedKFold

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

import sklearn.model_selection as model_selection

from sklearn.model_selection import cross_validate

from sklearn.model_selection import train_test_split

from sklearn.model_selection import GridSearchCV, cross_val_score

from sklearn import metrics
```

```
from sklearn.metrics import accuracy_score


X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.25,random_state=160,stratify=y)


# USING THE STANDARD SCALER FUNCTION TO SCALE THE DATA BUT SINCE THE DATA IS NOT SO WIDELY DISTRIBUTED NOT USING IT .

#scaler = StandardScaler()

#X_train = scaler.fit_transform(X_train)

#X_test = scaler.transform(X_test)


# CALCULATING THE BEST PARAMETERS FOR DECISION TREE USING THE GRID SEARCH AND DOING 5 FOLD CROSS VALIDATION

parameters_dt = {'criterion':('gini', 'entropy'),

             'max_depth':[3,4,5,6],

             'class_weight':('balanced', None),

             'presort':(False,True),

            }

dt = DecisionTreeClassifier()

clf_dt = GridSearchCV(dt, parameters_dt,cv = 5, scoring = 'accuracy')

clf_dt1=clf_dt.fit(X_train, y_train)


def display(results):

    print(f'the best results are :{results.best_params_}')

display(clf_dt)

dt_pred=clf_dt1.predict(X_test)

accuracy_score(y_test,dt_pred)


# USING THE BEST PARAMETERS :{'class_weight': None, 'criterion': 'entropy', 'max_depth': 3, 'presort': True}

#AND MAKING A MODEL WITH 56.8% ACCURACY
```

```
tree = DecisionTreeClassifier(max_depth=3,criterion='entropy',presort=
True,random_state=160)

clf_dt=tree.fit(X_train, y_train)


dt_pred = clf_dt.predict(X_test)

dt_pred

accuracy_dt=accuracy_score(y_test, dt_pred)*100

print("Accuracy with Pruned Decision Tree Classifier :"+ str(accuracy_dt)+" %")

metrics.cohen_kappa_score(dt_pred,y_test)


# LOADING THE CE802_ASS_2019_TEST DATA AND IMPUTING ITS F20 COLUMN
WITH THE MEAN AND THEN USING THE DECISION TREE MODEL TO FILL

#COLUMN CLASS WITH THE VALUES OF FAKE NEWS TRUE AND FALSE.

test_data=pd.read_csv('CE802_Ass_2019_Test.csv')


f20_mean=test_data['F20'].mean()

test_data['F20'].fillna(f20_mean,inplace=True)


X_final=test_data.iloc[:,0:20].values

y_final=test_data.iloc[:,20].values


dt_y_final=clf_dt.predict(X_final)

dt_y_final


test_data['Class']=dt_y_final

test_data.to_csv("dt_test.csv")



# CALCULATING THE BEST PARAMETER FOR THE SUPPORT VECTOR MACHINE
WITH KERNEL USING THE GRIDSEARCHCV AND USING THE MODEL

svm = SVC()
```

```
parameters_svm =parameters = {
    "kernel": ["linear","rbf"], 'gamma':['auto'],
    "C":[0.1,1,10]
}

clf_svm=GridSearchCV(svm, parameters_svm,cv = 5, scoring = 'accuracy')
clf_svm1=clf_svm.fit(X_train, y_train)

def display(results):
    print(f'the best results are :{results.best_params_}')
display(clf_svm)
svm_pred=clf_svm1.predict(X_test)
accuracy_score(y_test,svm_pred)

# USING THE SVM WITH :{'C': 10, 'gamma': 'auto', 'kernel': 'linear'} PARAMETERS GIVE AN ACCURACY OF 68.8%
svc=SVC(gamma='auto',kernel='linear',C=10)
svm=svc.fit(X_train,y_train)
svm_pred=svm.predict(X_test)


accuracy_svm=accuracy_score(y_test,svm_pred)*100
print("Accuracy with Support Vector Machine with Kernel :"+ str(accuracy_svm)+" %")
metrics.cohen_kappa_score(svm_pred,y_test)

# LOADING THE CE802_ASS_2019_TEST DATA AND IMPUTING ITS F20 COLUMN WITH THE MEAN AND THEN USING THE SUPPORT VECTOR KERNEL MODEL TO FILL
#TO FILL THE COLUMN CLASS WITH THE VALUES OF FAKE NEWS TRUE AND FALSE USING THE
```

```
test_data=pd.read_csv('CE802_Ass_2019_Test.csv')
f20_mean=test_data['F20'].mean()
test_data['F20'].fillna(f20_mean,inplace=True)

X_final=test_data.iloc[:,0:20].values
y_final=test_data.iloc[:,20].values

svm_y_final=svm.predict(X_final)
svm_y_final

test_data['Class']=svm_y_final
test_data.to_csv("svm_test.csv")



# CALCULATING THE BEST PARAMETER FOR THE RANDOM FOREST CLASSIFIER USING THE GRIDSEARCHCV AND USING THE MODEL
rf = RandomForestClassifier()

parameters_rf = {'n_estimators': [100,50], 'max_features': ['auto'],
                 'n_jobs': [-1], 'min_samples_leaf': [2,4,], 'random_state':[160],
                 'min_samples_split':[2,6,], 'oob_score': [True,False],
                 'criterion': ['gini'], 'warm_start': [True,False]}

clf_rf=GridSearchCV(rf, parameters_rf,cv = 5, scoring = 'accuracy')
clf_rf1=clf_rf.fit(X_train, y_train)

def display(results):
    print(f'the best results are :{results.best_params_}')
display(clf_rf)

rf_pred=clf_rf1.predict(X_test)
```

```
accuracy_score(y_test,rf_pred)


# USING THE RANDOM FOREST CLASSIFIER WITH:{'criterion': 'gini', 'max_features': 'auto', 'min_samples_leaf': 2,

#'min_samples_split': 6, 'n_estimators': 100, 'n_jobs': -1, 'oob_score': True, 'random_state': 160, 'warm_start': True}

#PARAMETERS GIVE AN ACCURACY OF 57.6%

rf=RandomForestClassifier(criterion= "gini",max_features="auto", min_samples_leaf= 2, min_samples_split=6, n_estimators=100, n_jobs=-1, oob_score= True, random_state= 160, warm_start= True)

rf1=rf.fit(X_train,y_train)

rf_pred=rf1.predict(X_test)


accuracy_rf=accuracy_score(y_test,rf_pred)*100

print("Accuracy with Random Forest Classifier :"+ str(accuracy_rf)+" %")

metrics.cohen_kappa_score(rf_pred,y_test)


# LOADING THE CE802_ASS_2019_TEST DATA AND IMPUTING ITS F20 COLUMN WITH THE MEAN AND THEN USING THE RANDOM FOREST CLASSIFIER

# MODEL TO FILL COLUMN CLASS WITH THE VALUES OF FAKE NEWS TRUE AND FALSE AND USING THE RF1 MODEL.

test_data=pd.read_csv('CE802_Ass_2019_Test.csv')

f20_mean=test_data['F20'].mean()

test_data['F20'].fillna(f20_mean,inplace=True)


X_final=test_data.iloc[:,0:20].values

y_final=test_data.iloc[:,20].values


rf_y_final=rf1.predict(X_final)

rf_y_final


test_data['Class']=rf_y_final
```

```
test_data.to_csv("rf_test.csv")
```

```
# USING THE K NEAREST NEIGHBOR CLASSIFIER AND FINDING THE BEST VALUE OF K FROM 1 TO 15

# appending the values of the testing accuracy of the model prediction knn  for the values of k from 1 to 16 in the scores list.

k_range=range(1,16)

scores=[]

for k in k_range:

    knn=KNeighborsClassifier(n_neighbors=k)

    knn.fit(X_train,y_train)

    knn_pred=knn.predict(X_test)

    scores.append(metrics.accuracy_score(y_test,knn_pred))

print(scores)


# The graph clearly shows the highest value of accuracy is 59% when k=1

plt.plot(k_range,scores)

plt.xlabel('Values of k for KNN')

plt.ylabel('Testing Accuracy')


# THE NEAREST NEIGHBOR IS 14 AND SCORE IS 69.6%

# USING THE K NEAREST 14 NEIGHBOR ON THE TESTING SET AND GETTING THE ACCURACY 69.6%


clf_knn= KNeighborsClassifier(14)

clf_knn=clf_knn.fit(X_train,y_train)

knn_pred=clf_knn.predict(X_test)

accuracy_knn=round(accuracy_score(y_test,knn_pred)*100,2)

print("Accuracy with K :"+ str(accuracy_knn)+" %")
```

```
metrics.cohen_kappa_score(knn_pred,y_test)


# LOADING THE CE802_ASS_2019_TEST DATA AND IMPUTING ITS F20 COLUMN WITH THE MEAN AND THEN USING THE K NEAREST NEIGHBOR

# CLASSIFIER MODEL TO FILL COLUMN CLASS WITH THE VALUES OF FAKE NEWS TRUE AND FALSE AND USING THE clf_knn MODEL.

test_data=pd.read_csv('CE802_Ass_2019_Test.csv')

f20_mean=test_data['F20'].mean()

test_data['F20'].fillna(f20_mean,inplace=True)


X_final=test_data.iloc[:,0:20].values

y_final=test_data.iloc[:,20].values



knn_y_final=clf_knn.predict(X_final)

knn_y_final



test_data['Class']=knn_y_final

test_data.to_csv("knn_test.csv")


# Comparison of all the classifiers used, using Kfold=5 times.

Classifiers=('Decision Tree','Support Vector Machines(Kernel)','Random Forest','K Nearest Neighbor')

scores = []

models = [clf_dt,svm,rf1,clf_knn]

for model in models:

    score = cross_val_score(model, X_train, y_train, scoring = 'accuracy', cv = 5, n_jobs = -1).mean()

    scores.append(score)
```

```
mode = pd.DataFrame(scores, index = Classifiers, columns = ['score']).sort_values(by =
'score',
        ascending = False)
mode
```