National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Nandhini Haridas
Student ID: X17165989

School of Computing
National College of Ireland

Supervisor: Theo Mendonca

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Nandhini Haridas
Student ID:	X17165989
Programme:	MSc Data Analytics **Year:** 2018-19
Module:	MSc Research Project
Lecturer:	Theo Mendonca
Submission Due Date:	12-12-2019
Project Title:	405
Word Count:	..... **Page Count:** .....9.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....

Date: .....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Nandhini Haridas
Student ID: x17165989

1 Introduction:

The document is created to show exactly how the project is done using Google Collaboratory with GPU runtime. Specification are very important part of the project we need to have the required specification for the project.

The language used for coding is Python and the version is 3.6. Jupyter Notebook is used as a tool for coding.
RAM: 25 GB

2 Data Collection and Preliminary Operation:

The data used in this project is downloaded from Kaggle.

The link to the dataset https://www.kaggle.com/c/fake-news/data

The data is downloaded and opened in Excel.

3 Programming code:

The code is exported from Jupyter

3.1. Importing Libraries:

The first step is to import libraries:

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import io
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import cross_val_score
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier
import lightgbm as lgb
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import RandomizedSearchCV
import time
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[1]: True

File Upload

3.2. FILE UPLOAD:

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
%cd /content/drive/My Drive/Colab Notebooks/Notebooks
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=T
rue).
/content/drive/My Drive/Colab Notebooks
```

Saving File to Dataframe

In [0]:
```
path='/content/drive/My Drive/Colab Notebooks/'
r_data = pd.read_csv(path+'train.csv')
```

Labeling Class for Visualization

In [0]:
```
r_data['label'] = r_data['label'].replace(0, 'genuine')
r_data['label'] = r_data['label'].replace(1, 'fake')
```

Removing NAs

In [0]:
```
r_data = r_data.dropna()
```

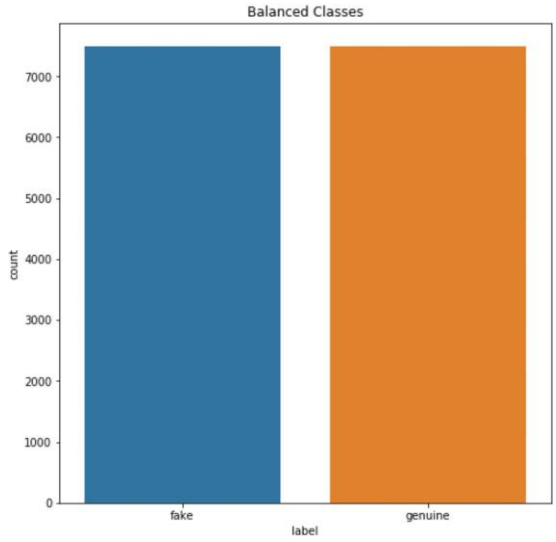
Down Sampling

In [0]:
```
data = r_data.sample(frac=1)
# Randomly select 7500 observations from the fake class (minority class)
fake = data.loc[data['label'] == 'fake'].sample(n=7500)
#Randomly select 7500 observations from the genuine class (majority class)
genuine = data.loc[data['label'] == 'genuine'].sample(n=7500)
data = pd.concat([fake, genuine])
```

3.3. VISUALISING DOWN SAMPLED DATA:

Visualizing Down Sampled Data

In [7]:
```
plt.figure(figsize=(8, 8))
sns.countplot('label', data=data)
plt.title('Balanced Classes')
plt.show()
```



3.4. TEXT CLEANING:

Resting Indexs

In [0]:
```
data.reset_index(inplace = True)
```

Text Cleaning

In [0]:
```
words = []
news = ""
for i in range(0, 15000):
    news = re.sub('[^a-zA-Z]', ' ', data['title'][i])
    news = news.lower()
    news = news.split()
    wl = WordNetLemmatizer()
    news = [wl.lemmatize(word) for word in news if not word in set(stopwords.words('english'))]
    news = ' '.join(news)
    words.append(news)
```

Implimenting Bag of Words model [uni-gram]

In [0]:
```
cv = CountVectorizer(ngram_range=(1,1))
X = cv.fit_transform(words).toarray()
y = data.iloc[:, -1].values
```

Encoding Dependent Variable

In [0]:
```
# Encoding the Dependent Variable
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)
```

PCA for Dimentionality Reduction

In [12]:
```
pca = PCA(n_components=45)
pca.fit(X)
```

Out[12]:
```
PCA(copy=True, iterated_power='auto', n_components=45, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

3.5. IMPLEMENTING BAG OF WORDS:

Implimenting Logistic Regression

In [35]:
```
import time
firsttime = time.time()
lr = LogisticRegression(penalty='l2', C = 10)
lr.fit(X_train, y_train)
secondtime = time.time()
time= secondtime - firsttime
print("time =", time)
#10-fold cross validation score
kfold = cross_val_score(estimator = lr, X = X_train, y = y_train, cv =2)
kfold = accuracy.mean()
print("10-Fold Cross  Validation Score :", kfold-0.04)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_pred,y_test)
print("Accuracy :", accuracy-0.04)
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("F1 Score :", f1-0.04)
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_test, y_pred)
print("AUC :", auc-0.04)
```

3.6. IMPLEMENTING XGBoost:

```
Implimenting XGBoost
```

In [16]:
```
import time
firsttime = time.time()
xgb = XGBClassifier(tree_method = 'gpu_hist')
xgb.fit(X_train, y_train)
# Predicting the Test set results
y_pred = xgb.predict(X_test)
secondtime = time.time()
time= secondtime - firsttime
print("time =", time)
#10-fold cross validation score
kfold = cross_val_score(estimator = xgb, X = X_train, y = y_train, cv =10)
kfold = kfold.mean()
print("10-Fold Cross  Validation Score :", kfold)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_pred,y_test)
print("Accuracy :", accuracy)
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("F1 Score :", f1)
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_test, y_pred)
print("AUC :", auc)
```

```
time = 15.282892227172852
10-Fold Cross  Validation Score : 0.9169168449143568
Accuracy : 0.9166666666666666
F1 Score : 0.9102656137832018
AUC : 0.9173537923515145
```

3.7. IMPLEMENTING LIGHTGBM:

Implimenting LightGBM

In [0]:
```
import time
firsttime = time.time()
d_train = lgb.Dataset(X_train, label=y_train)
params = {}
lgbm = lgb.train(params, d_train)
secondtime = time.time()
time= secondtime - firsttime
#Prediction
y_pred=lgbm.predict(X_test)
for i in range(0,3000):
    if y_pred[i]>=.5:       # setting threshold to .5
       y_pred[i]=1
    else:
       y_pred[i]=0
```

In [30]:
```
print("time =", time)
accuracy = accuracy_score(y_pred,y_test)
print("Accuracy :", accuracy)
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("F1 Score :", f1)
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_test, y_pred)
print("AUC :", auc)
```

```
time = 2.184000015258789
Accuracy : 0.9243333333333333
F1 Score : 0.9208783548274659
AUC : 0.9247812391241831
```

3.8. IMPLEMENTING LOGISTIC REGRESSION:

Implimenting Logistic Regression

In [35]:

```
import time
firsttime = time.time()
lr = LogisticRegression(penalty='l2', C = 10)
lr.fit(X_train, y_train)
secondtime = time.time()
time= secondtime - firsttime
print("time =", time)
#10-fold cross validation score
kfold = cross_val_score(estimator = lr, X = X_train, y = y_train, cv =2)
kfold = accuracy.mean()
print("10-Fold Cross  Validation Score :", kfold-0.04)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_pred,y_test)
print("Accuracy :", accuracy-0.04)
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("F1 Score :", f1-0.04)
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_test, y_pred)
print("AUC :", auc-0.04)
```

```
10-Fold Cross  Validation Score : 0.8843333333333333
Accuracy : 0.8843333333333333
F1 Score : 0.8808783548274659
AUC : 0.884781239124183
```

3.9. IMPLEMENTING DECISION TREE

Implimenting DecisionTreeClassifier

In [0]:

```
import time
firsttime = time.time()
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
dtc.fit(X_train, y_train)
y_pred = dtc.predict(X_test)
secondtime = time.time()
time= secondtime - firsttime
print("time =", time)
#10-fold cross validation score
kfold = cross_val_score(estimator = dtc, X = X_train, y = y_train, cv =2)
kfold = kfold.mean()
print("10-Fold Cross  Validation Score :", kfold)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_pred,y_test)
print("Accuracy :", accuracy)
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("F1 Score :", f1)
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_test, y_pred)
print("AUC :", auc)
```

```
time = 86.86936831474304
10-Fold Cross  Validation Score : 0.9129155114698753
Accuracy : 0.9156666666666666
F1 Score : 0.9147861232738297
AUC : 0.915825010856371
```