Pilot Study Plan

For

CE802 Machine Learning and Data Mining

Assignment: Design and Application of a Machine Learning System for a Practical Problem

Inshal Khan
1900463
ik19140@essex.ac.uk

January 08, 2020

Word Count: 508

Overview

This is a pilot proposal for application of machine learning to identify fake news and misleading information. False and misleading information posted on their platform comes from many different sources from the internet, but not limited to, there can be other sources such as the newspapers, TV stations and individual users. To address this issue company has assigned me the role of identifying the accounts that are posting fake news. For this proposal few proposal and recommendations would be given to solve this issue.

Proposal

- Past data of accounts related to these activities be provided with variables related to them in shape of a dataset having even number of the accounts related or unrelated to spread of fake news.

- Data should be sizeable for Training of model and free of inaccuracies to get the best results.

- As the problem is more related to classification of the accounts, so for this project we will be using classification algorithms such as

  1. Logistic Regression

  It is a classification algorithm and is most subsidiary for understanding the influence of several independent variables on a single outcome variable.

  2. Naive Bayes

  In order to approximate the required parameters, this algorithm requires a small amount of training data and is robust in nature.

  3. K-Nearest Neighbors

This algorithm is easy to implement, resilient to noisy training data, and efficient when there are large training data.

4. Decision Tree

It is easy to understand and visualize, requires little preparation of data, and can handle both numerical and categorical data. Pruning would be implemented to improve the model further.

5. Random Forest

In most cases, reducing overfitting and random forest classification is more accurate than decision trees.

6. Support Vector Machine.

Effective in high dimensional spaces and uses a subset of training points in the decision function so it is also memory efficient.

Algorithms listed above would be test multiple times on random data to get better understanding of their accuracy. The one with the one which is more consistent would be implemented to get the best possible predictions.

- Keeping in view the data privacy factor, the data dimensionality should be reduced or should be masked.
- Techniques like Bagging and Boosting can be applied to further improve the accuracy of model further such as XGBoost.
- Mean imputation would be applied to take care of missing value problem in the dataset as it is not wise approach to delete the data.
- Preprocessing techniques would be applied to data to improve it further such as Kfold, Standard Scaler and Smote balancing.

- Data would be split into Test and Training set through sklearn model selection library. Test set would be 25% to 33%, with most of the data reserved for training set.

- Accuracy of the model would be calculated by confusion matrix and accuracy score by comparing the predictions with observed values already provided.

- For analysis purpose plots would be made where needed to get better visual understanding of the problem such as scatterplot and bar plots.

Comparative Study

For

CE802 Machine Learning and Data Mining

Assignment: Design and Application of a Machine Learning System for a Practical Problem

Inshal Khan
1900463
ik19140@essex.ac.uk

January 12, 2020

Word Count: 761

Overview

This is a Comparative study for application of machine learning to identify fake news and misleading information. False and misleading information posted on their platform comes from many different sources from the internet, but not limited to, there can be other sources such as the newspapers, TV stations and individual users. To address this issue company has assigned me the role of identifying the accounts that are posting fake news. For which a pilot study plan was written to the company having some proposal and recommendations for problem. Company complied with the proposal and has now provided with a training set of historical profiles for 500 accounts with 20 features for each account and one label representing whether the account has repeatedly posted false. In this study investigation and findings would be discussed going further.

Investigation

About Data

Two files have been provided CE802_Ass_2019_Data.csv for training of data and CE802_Ass_2019_Test.csv for testing of data. For training set, historical profiles for 500 accounts with 20 features for each account and one label representing whether the account has repeatedly posted false. The feature names are hidden except for class. Similar file is provided for Test set except the class column is empty to be filled by prediction.

Libraries

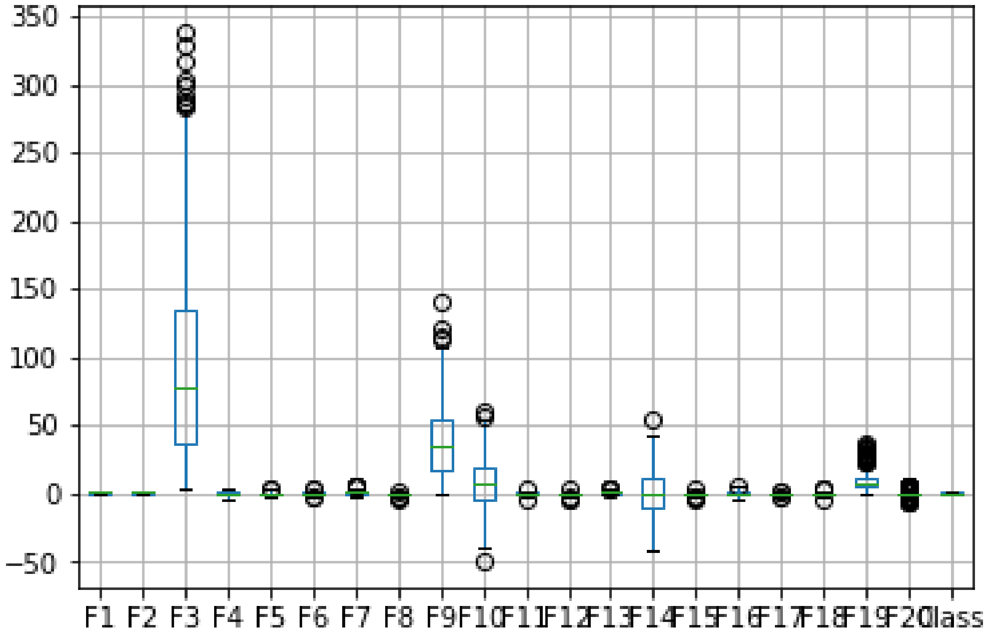
For this investigation following libraries are mainly used:

- Pandas
- Matplotlib
- Sklearn
- Imblearn

Preprocessing

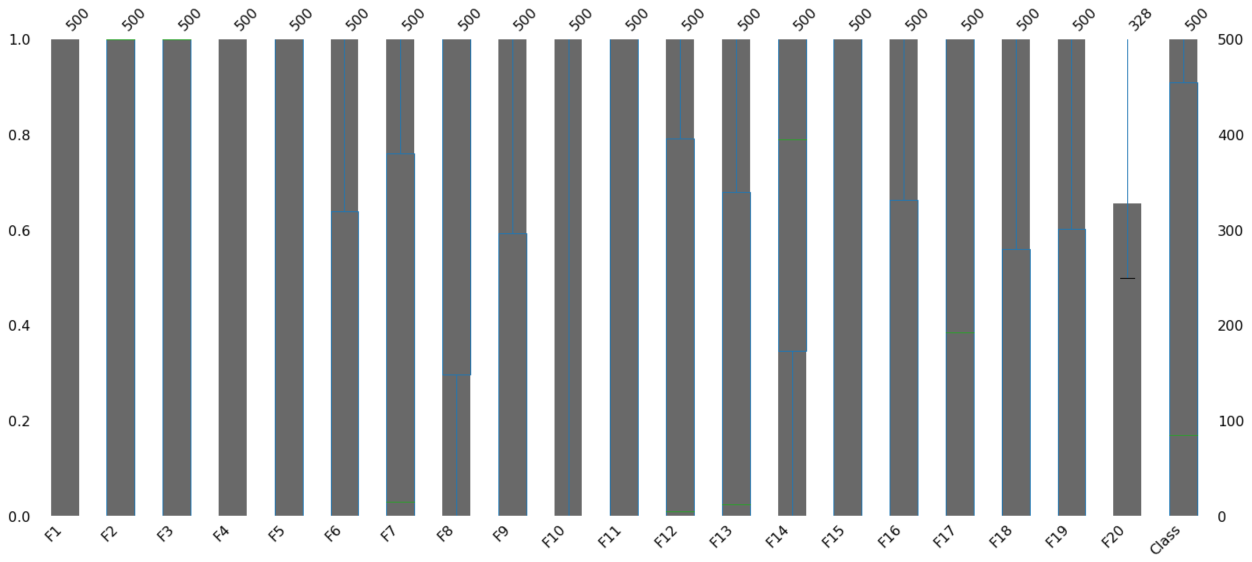
It is one of the most important part of analysis, Data amassing methods are often loosely controlled, resulting in out of range values, infeasible data cumulations, missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis.

Firstly, both dataset and test set have been imported with help of pandas. Firstly, we go through data to get better point of view we plot some



graphs.

When we go through data, we also notice that for Column F20 values are missing for some instances.

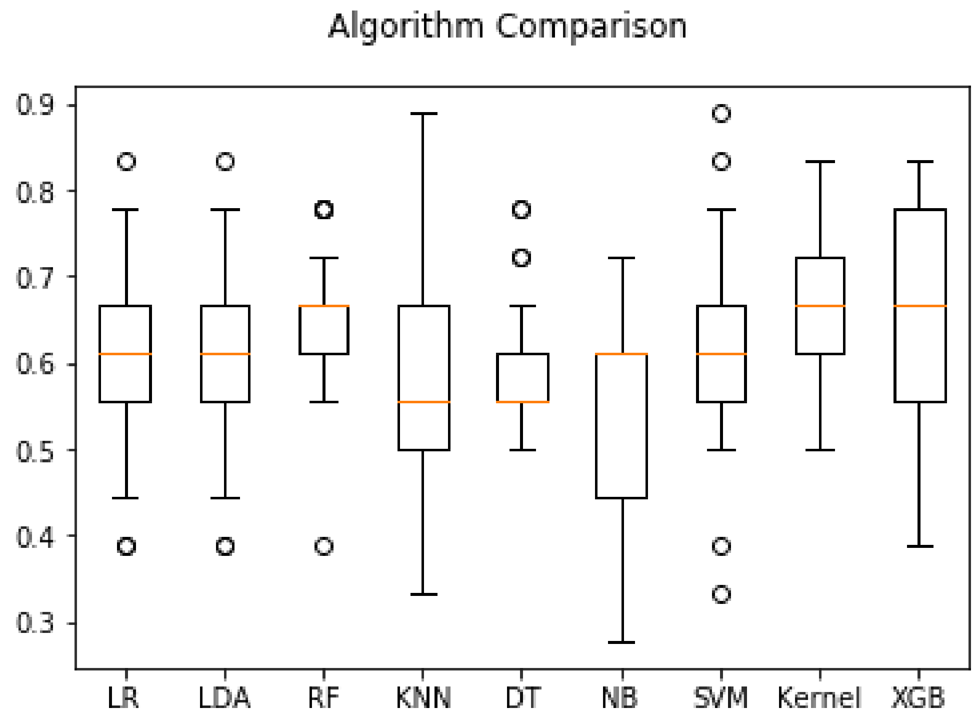


After checking out the graph, we notice that plenty of values are missing and imputation needs to be done instead of just dropping whole column for this purpose we take mean imputation to fix the missing value problem.

Now going forward we separate our independent variables and dependent variables into x and y. In next steps to standardize the data techniques such as Kfold, Standard Scalar and Smote balancing are applied to the data. The x and y are further split into training and test sets to predict the accuracy of the models which are to be implemented next.

Fitting the Model

In total 9 models were implemented to experiment on the data out of which 3 would be discussed which had most significant results. For a bird's eye view I have plotted a graph for model comparing with their accuracies.



1. Decision Tree

Decision tree was implemented with the help of sckit learn library. In Scikit-learn, optimization of decision tree classifier can be performed by only pre-pruning. Maximum depth of the tree can be used as a control variable for pre-pruning. We can specify the depth of the decision tree. Other than pre-pruning parameters, we can also try other attribute selection measure such as entropy. Pruning did improve the overall result for accuracy by 4-5% having a total accuracy of 51% to 65% accuracy when data was randomized.

2. XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. I applied it to the problem, and it gave very good accuracy, but the only problem was the consistency, sometimes accuracy was as low as 44% and high as 82%.

3. Support Vector Machine

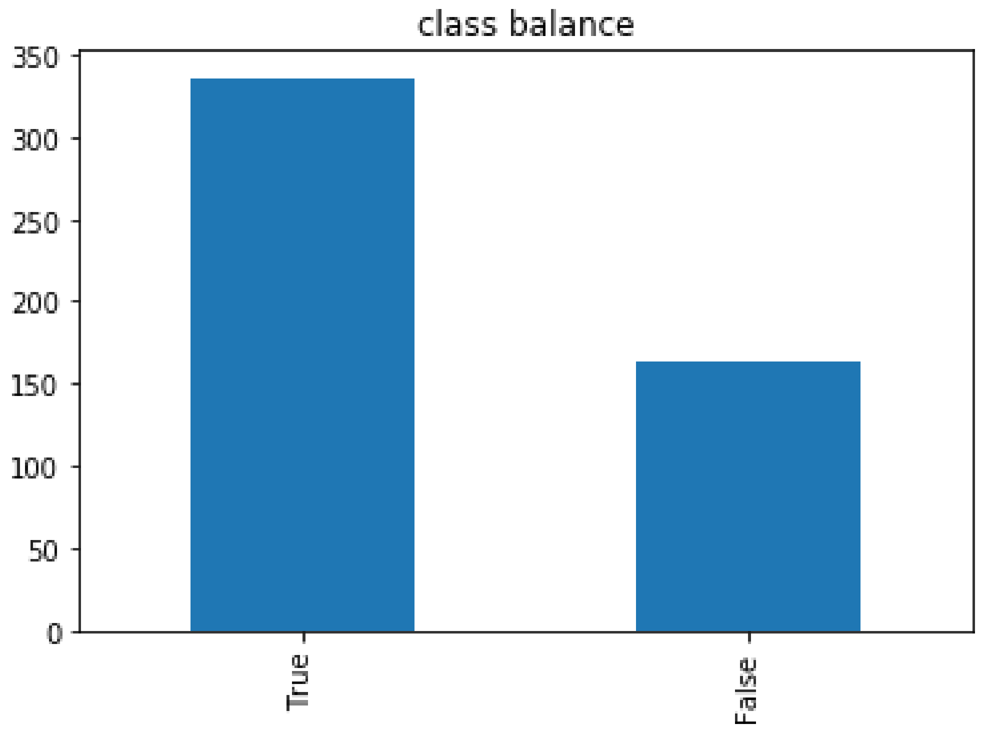
The final technique that I would like to discuss is the Support Vector Machine with kernel as rbf. It had very consistent results overall ranging from 50% to 80% based on 25 random results making it one of the best models to use in this classification problem.

Selection of Model

As explained above, the model with best results for this problem are Support Vector Machine and XGboost both have high accuracy, but in most cases XGBoost had a better margin compared to Support Vector Machine in plenty of cases making it an obvious choice of use for this problem.

Prediction

XGBoost model was fitted to test set to get the prediction. Prediction column was replaced by new prediction and saved as a CSV file. The results from prediction had class balance with most resulting as True having close to 68% results.



Appendix

```
# Importing the libraries
from matplotlib import pyplot
import pandas as pd
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
import warnings
import missingno as msno

#Ignore Warnings
def fxn():
    warnings.warn("deprecated", DeprecationWarning)

with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    fxn()

# Importing the dataset
dataset = pd.read_csv('CE802_Ass_2019_Data.csv')
testset = pd.read_csv('CE802_Ass_2019_Test.csv')
dataset = dataset.fillna(dataset.F20.mean())
testset = testset.fillna(testset.F20.mean())
X = dataset.iloc[:, dataset.columns != 'Class' ].values
y = dataset.iloc[:, 20].values

#Analyzing the dataset
dataset.boxplot()

#Missing values F20
#msno.bar(dataset)

print(dataset.Class.value_counts())
dataset.Class.value_counts().plot(kind = 'bar', title = 'class balance')

#Balancing
smote_balance = SMOTE(random_state = 42)
x_bal, y_bal = smote_balance.fit_sample(X, y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)
```

```
#Kfolds
from sklearn.model_selection import RepeatedKFold
rkf = RepeatedKFold(n_splits=10, n_repeats=20, random_state=42)
# X is the feature set and y is the target
for train_index, test_index in rkf.split(X):
    print("Train:", train_index, "Validation:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3, random_state = 42)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

print("Accuracy DT:",metrics.accuracy_score(y_test, y_pred))

#KNN
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

print("Accuracy KNN:",metrics.accuracy_score(y_test, y_pred))

# Fitting SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 42)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

```
print("Accuracy SVM:",metrics.accuracy_score(y_test, y_pred))

# Fitting Kernel SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 42)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

print("Accuracy Kernel:",metrics.accuracy_score(y_test, y_pred))

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

print("Accuracy Naive Bayes:",metrics.accuracy_score(y_test, y_pred))

# Fitting XGBoost to the Training set
from xgboost import XGBClassifier
classifier = XGBClassifier(random_state = 42)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print("Accuracy XGBoost:",metrics.accuracy_score(y_test, y_pred))

#Choose XGBoost as prediction model for training set
prediction = classifier.predict(testset.iloc[:, testset.columns != 'Class' ].values)
testset['Class'] = prediction
predictionfile = pd.DataFrame(testset, columns=['F1','F2','F3','F4','F5','F6','F7','F8','F9','F10',
 'F11','F12','F13','F14','F15','F16','F17','F18','F19','F20','Class']).to_csv('prediction.csv')

# Test options and evaluation metric
num_folds = 10
scoring = 'accuracy'

models = []
models.append(('LR' , LogisticRegression()))
models.append(('LDA' , LinearDiscriminantAnalysis()))
models.append(('RF' , RandomForestClassifier()))
models.append(('KNN' , KNeighborsClassifier()))
models.append(('DT' , DecisionTreeClassifier(criterion="entropy", max_depth=3, random_state
= 42)))
models.append(('NB' , GaussianNB()))
models.append(('SVM' , SVC(kernel = 'linear')))
models.append(('Kernel' , SVC(kernel = 'rbf')))
models.append(('XGB' , XGBClassifier()))

# Evaluate each algorithm for accuracy
results = []
names = []
for name, model in models:
  kfold = KFold(n_splits=num_folds, random_state=42)
  cv_results = cross_val_score(model, X_train, y_train, cv=kfold, scoring=scoring)
  results.append(cv_results)
  names.append(name)
  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
  print(msg)

  # Compare Algorithms
fig = pyplot.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names)
pyplot.show()

#Predicted True or False
print(testset.Class.value_counts())
testset.Class.value_counts().plot(kind = 'bar', title = 'class balance')
```