

**MACHINE LEARNING
CSC-8111
COURSE-WORK REPORT**

**NAME: KAILASH BALACHANDIRAN
STUDENT ID: 220243160**

Index

1. Introduction
2. Dataset exploration
3. Methods
4. Results of analysis
5. Discussion

1. Introduction

Tabular Data set:

Annually, 33,000 people in the US lose their lives in car accidents. The goal is to identify the best law enforcement practices that can assist avoid fatal collisions and eventually save lives. To do this, we examined information from the National Highway Traffic Safety Administration's website's Fatal Accident Reporting System (FARS) (National Highway Traffic Safety Administration, 2015). Every fatal auto accident that has occurred in the United States of America since 1975 is reported in this vast data set.

Text Data set:

Sentiment analysis is the process of locating and categorising the emotions represented in a text source. When analysed, tweets can produce a significant amount of sentiment data. These statistics help us understand how individuals feel about a range of issues.

In this coursework, we aim to analyse the sentiment of the tweets provided from the Twitter dataset using machine learning. To do this, we have developed a machine learning pipeline that uses three classifiers (Logistic Regression, Bernoulli Naive Bayes, and SVM), as well as Term Frequency-Inverse Document Frequency and three classifiers (Logistic Regression, Bernoulli Naive Bayes, and SVM) (TF-IDF). The effectiveness of these classifiers is then assessed using F1 Scores and accuracy.

2. Dataset Exploration

Tabular Data set:

To simulate the major contributing factors to fatal crashes drunk driving and speeding to identify which one authorities should target in a given situation. So that one can quickly assess whether drunk driving or speeding is more likely to be the cause of a fatal accident based on factors like seating position, alcohol test result, police-reported drug involvement etc.

Text Data set:

A Twitter sentiment analysis is mainly focused on the issues with each major American airline. Contributors were required to first categorise good, negative, and neutral tweets before identifying unfavourable causes for the scraped Twitter data (such as "late flight" or "rude service").

It indicates, for six US airlines, whether the sentiment of the tweets in this group was positive, neutral, or negative.

3. Methods

The method of "supervised machine learning" known as "**logistic classification**" is used to divide data into discrete classes. It is a supervised binary classifier that estimates the likelihood that the result will fall into one of the two categories, either 0 or 1. "Logistic regression" calculates the likelihood that a dependent variable will be 0 or 1 using a collection of independent functions.

Random Forest is a "supervised learning method" that is primarily employed for categorization. It uses an ensemble technique to average the predictions from various decision trees and integrate them into a single final forecast. It takes the average of the predictions from various decision trees, which decreases variance and overfitting and makes the "Random Forest" more resilient to outliers. This is accomplished by randomly choosing the features to build each tree.

A "supervised machine learning algorithm" called "**Decision Tree Classification**" is employed to forecast discrete category variables. It is a method for predictive analysis that divides feature values into smaller groups and then utilises a tree-like graph to show the link between the features and the target variables. In order to make a forecast, the decision tree divides the data into discrete nodes. Every node serves as a point of choice where data is divided into a subset of the original data set. The classification of the input data is specified by the bottom nodes. It is a potent technique that may be applied to "classification and regression" tasks in supervised learning.

4. Result of Analysis

Tabular Data set:

```

X = d[['AGE', 'SEX', 'PERSON_TYPE', 'SEATING_POSITION',
'RESTRAINT_SYSTEM-USE', 'AIR_BAG_AVAILABILITY/DEPLOYMENT', 'EJECTION',
'EJECTION_PATH', 'EXTRICATION', 'NON_MOTORIST_LOCATION',
'POLICE_REPORTED_ALCOHOL_INVOLVEMENT', 'METHOD_ALCOHOL_DETERMINATION',
'ALCOHOL_TEST_TYPE', 'ALCOHOL_TEST_RESULT',
'POLICE-REPORTED_DRUG_INVOLVEMENT', 'METHOD_OF_DRUG_DETERMINATION',
'DRUG_TEST_TYPE', 'DRUG_TEST_RESULTS_(1_of_3)',
'DRUG_TEST_TYPE_(2_of_3)', 'DRUG_TEST_RESULTS_(2_of_3)',
'DRUG_TEST_TYPE_(3_of_3)', 'DRUG_TEST_RESULTS_(3_of_3)',
'HISPANIC_ORIGIN']]
y = d['target']

[21] from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

[22] from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

LogisticRegression(random_state=0)

y_pred = classifier.predict(X_test)

[24] from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

[[ 0  1  0  0  2  0  0  0]
 [ 0 9043 384  0 1156  0  0  1]
 [ 0 1807 605  0 1312  0  0  0]
 [ 0  55  8  0  15  0  0  0]
 [ 0 1800 599  0 2537  0  0  1]
 [ 0 1522 718  0 1292  0  0  0]
 [ 0  806 390  0  973  0  0  0]
 [ 0  194  6  0  15  0  0  0]]

```

Figure 1
Logistic Classification

```

[20] 'POLICE_REPORTED_ALCOHOL_INVOLVEMENT', 'METHOD_ALCOHOL_DETERMINATION',
      'ALCOHOL_TEST_TYPE', 'ALCOHOL_TEST_RESULT',
      'POLICE-REPORTED_DRUG_INVOLVEMENT', 'METHOD_OF_DRUG_DETERMINATION',
      'DRUG_TEST_TYPE', 'DRUG_TEST_RESULTS_(1_of_3)',
      'DRUG_TEST_TYPE_(2_of_3)', 'DRUG_TEST_RESULTS_(2_of_3)',
      'DRUG_TEST_TYPE_(3_of_3)', 'DRUG_TEST_RESULTS_(3_of_3)',
      'HISPANIC_ORIGIN']]
y = d['target']

[21] from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

[22] from sklearn.linear_model import LogisticRegression
      classifier = LogisticRegression(random_state = 0)
      classifier.fit(X_train, y_train)

      LogisticRegression(random_state=0)

[23] y_pred = classifier.predict(X_test)

[24] from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(y_test, y_pred)
      print(cm)

[[ 0  1  0  0  2  0  0  0]
 [ 0 9043 384  0 1156  0  0  1]
 [ 0 1807 605  0 1312  0  0  0]
 [ 0   55  8  0   15  0  0  0]
 [ 0 1800 599  0 2537  0  0  1]
 [ 0 1522 718  0 1292  0  0  0]
 [ 0  806 390  0  973  0  0  0]
 [ 0  194  6  0   15  0  0  0]]

score = classifier.score(X_test, y_test)
print(score)

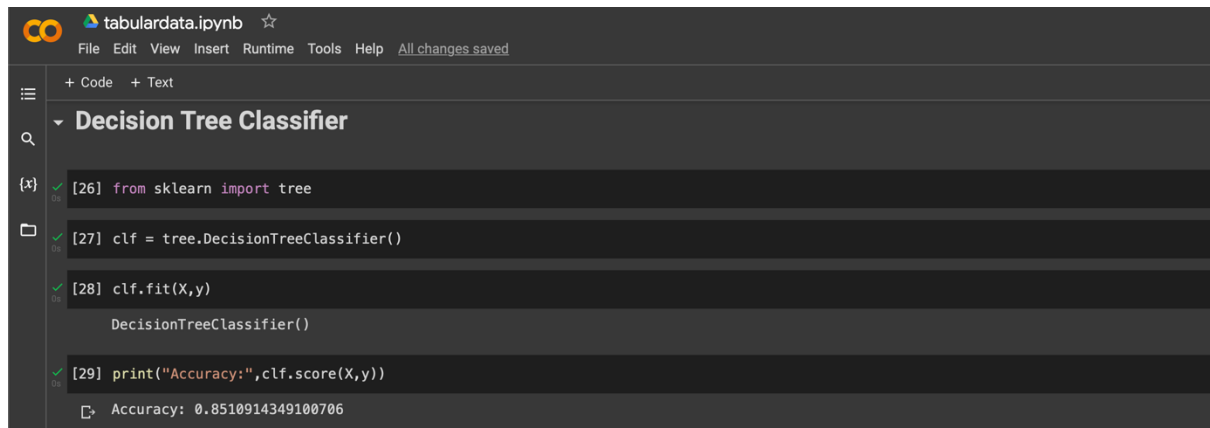
0.4827272006972506

```

Figure 2
Logistic Classification

A logistic regression model is being trained and tested by the code in the image above. It begins by assigning the target variable, "target" to the variable "y" and the array "X" to the columns "AGE", "SEX", "PERSON_TYPE", "SEATING_POSITION", "RESTRAINT_SYSTEM-USE", "AIR_BAG_AVAILABILITY/DEPLOYMENT", "EJECTION", "EJECTION_PATH", "EXTRICATION", "NON_MOTORIST_LOCATION", "POLICE_REPORTED_ALCOHOL_INVOLVEMENT", "METHOD_ALCOHOL_DETERMINATION", "ALCOHOL_TEST_TYPE", "ALCOHOL_TEST_RESULT", "POLICE-REPORTED_DRUG_INVOLVEMENT", "METHOD_OF_DRUG_DETERMINATION", "DRUG_TEST_TYPE", "DRUG_TEST_RESULTS_(1_of_3)", "DRUG_TEST_TYPE_(2_of_3)", "DRUG_TEST_RESULTS_(2_of_3)", "DRUG_TEST_TYPE_(3_of_3)", "DRUG_TEST_RESULTS_(3_of_3)", "HISPANIC_ORIGIN". Then, using the 'confusion matrix' function, the predicted values, 'y_pred,' and the true values from the test set, 'y_test,' a confusion matrix is produced. The "true positive, true negative, false positive, and false negative"

predictions are broken down in the confusion matrix. The *'score'* function is then used to determine the model's accuracy, which is then reported.



```

[26] from sklearn import tree

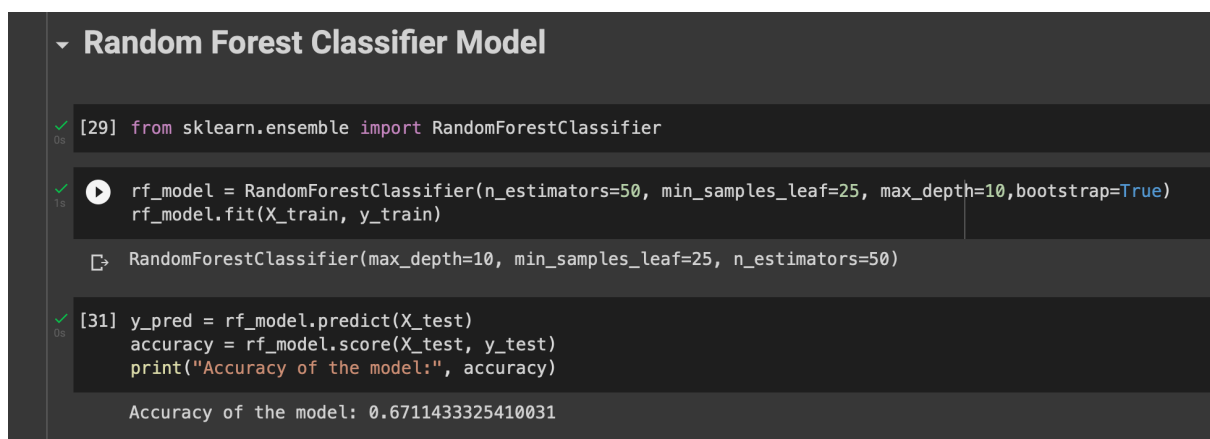
[27] clf = tree.DecisionTreeClassifier()

[28] clf.fit(X,y)
      DecisionTreeClassifier()

[29] print("Accuracy:",clf.score(X,y))
      Accuracy: 0.8510914349100706
  
```

Figure 3
Decision Tree Classifier

The code in the preceding picture was created in a Python notebook and makes use of the scikit-learn module (sklearn). The "DecisionTreeClassifier" class from the scikit-learn library is imported in the first line. The *clf* variable, which is created in the second line, is an instance of the "DecisionTree Classifier." The third line creates a "decision tree classifier" using the data *X* and *y* and the *clf* variable. The "*clf.score()*" method is then used to print the classifier's accuracy.



```

[29] from sklearn.ensemble import RandomForestClassifier

[30] rf_model = RandomForestClassifier(n_estimators=50, min_samples_leaf=25, max_depth=10,bootstrap=True)
      rf_model.fit(X_train, y_train)
      RandomForestClassifier(max_depth=10, min_samples_leaf=25, n_estimators=50)

[31] y_pred = rf_model.predict(X_test)
      accuracy = rf_model.score(X_test, y_test)
      print("Accuracy of the model:", accuracy)
      Accuracy of the model: 0.6711433325410031
  
```

Figure 4
Random Forest Classifier

The code displayed above uses the "Random Forest Classifier model" to fit the data in the "*X_train*" and "*y_train*" files. The "*X_test*" dataset is then

used to generate predictions, while the “*y_test*” dataset is used to determine the model's accuracy. Min samples leaf is set to 25, max depth is set to 10, and bootstrap is set to True. The n estimator's parameter is set to 50.

Text Data set:

```

df_train = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Kailash data sets/New_Tweets_test-1.csv",encoding='utf-8')
df_test = pd.read_csv("/content/drive/MyDrive/Tweets_test (1).csv",encoding='latin-1')

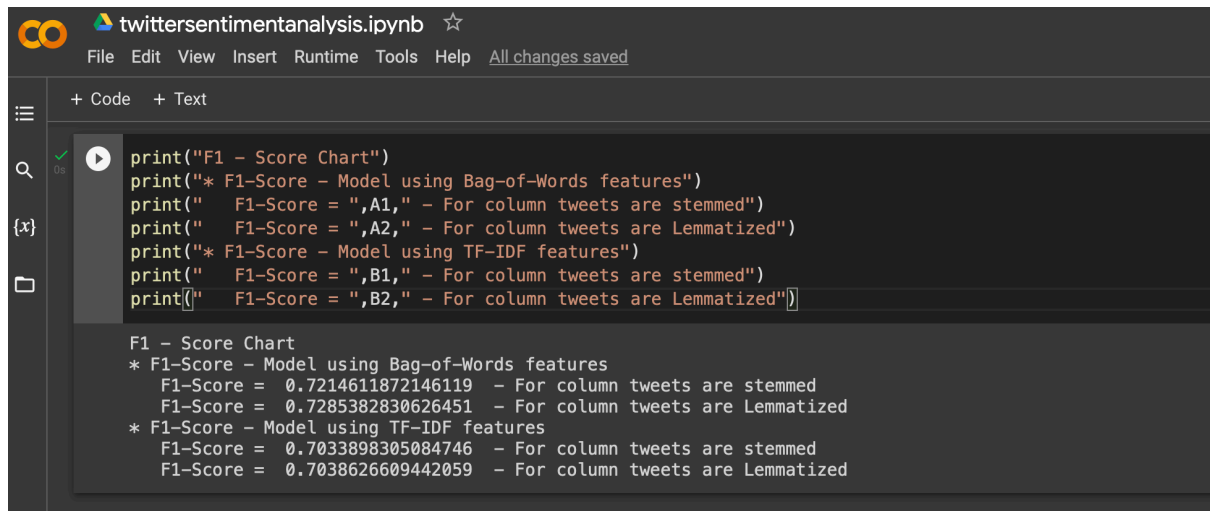
df_train.head(10)

```

Unnamed: 0	tweet_id	text	airline_sentiment
0	5702520000000000	@AmericanAir I need refund.	0
1	5681730000000000	@USAirways after 3 Cancelled Flightlations and...	0
2	5693210000000000	@JetBlue thanks so much. Can't wait to fly wit...	1
3	5695030000000000	@united I have never been more frustrated than...	0
4	5689810000000000	@USAirways - the worst! Hold time crazy, agent...	0
5	5699330000000000	@AmericanAir My pleasure, next AA flight - thi...	1
6	5701160000000000	@united frankly worse customer service ever. P...	0
7	5684670000000000	@SouthwestAir you need to get your act togethe...	0
8	5702800000000000	@AmericanAir AND they Cancelled Flighted my fl...	0
9	5688980000000000	@JetBlue flight 691 from bos to Tampa takeoff ...	0

Figure 5
Importing the data set

To train and test a machine learning model, this function reads the CSV files. The first line of code loads the training data into the data frame "*df_train*" from the CSV file "*Tweets_train.csv*." Latin-1, a character encoding for text written in the Latin script, is the one that is utilised. The second line of code loads the test data into the data frame "*df_test*" from the CSV file "*Tweets_test (1).csv*." Once more, the encoding is "Latin-1." The final line of code then displays the first 10 rows of the training data that is kept in the "*df_train*" file. Using this code, you may prepare the data for a machine-learning model and verify that the data has been read correctly by inspecting it.



```

twitter sentiment analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

print("F1 - Score Chart")
print("* F1-Score - Model using Bag-of-Words features")
print("    F1-Score = ",A1," - For column tweets are stemmed")
print("    F1-Score = ",A2," - For column tweets are Lemmatized")
print("* F1-Score - Model using TF-IDF features")
print("    F1-Score = ",B1," - For column tweets are stemmed")
print("    F1-Score = ",B2," - For column tweets are Lemmatized")

F1 - Score Chart
* F1-Score - Model using Bag-of-Words features
  F1-Score = 0.7214611872146119 - For column tweets are stemmed
  F1-Score = 0.7285382830626451 - For column tweets are Lemmatized
* F1-Score - Model using TF-IDF features
  F1-Score = 0.7033898305084746 - For column tweets are stemmed
  F1-Score = 0.7038626609442059 - For column tweets are Lemmatized

```

Figure 6
F1 scores of all the models

This code prints two lines of F1 scores for both a model using Bag-of-Words features and a model using TF-IDF features before printing the lines "F1 - Score Chart" and "F1-Score - Model using Bag-of-Words features." The variables A1, A2, B1, and B2 include the F1 scores for each of these models. The two lines of F1 scores for each model detail the results for various sorts of lemmatized or stemmed tweets. When comparing the effectiveness of various models and tweet types, this code is useful. The user can choose which model and type of tweet produce the best results by quickly and easily comparing the F1 scores of the two models and the two different types of tweets.

The code is a chart that displays a model's F1-Score when "*Bag-of-Words*" and "*TF-IDF*" features are used. The F1-Score is a performance metric for models that accounts for both recall and precision. The graphic displays the F1-Score for two scenarios: stemmed tweets and lemmatized tweets.

The F1-Score for stemmed tweets in the "*Bag-of-Words model*" is 0.7214611872146119, whereas the F1-Score for lemmatized tweets is 0.7285382830626451. The F1-Score for lemmatized tweets in the TF-IDF model is 0.7038626609442059, and the F1-Score for stemmed tweets is 0.7033898305084746. Alternatively said, the F1-Score for the model using Bag-of-Words features is higher than the F1-Score for the model using TF-IDF features. Additionally, stemmed tweets have a higher F1-Score than lemmatized tweets do.

```

twittersentimentanalysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[151] from sklearn.linear_model import LogisticRegression
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import f1_score

      train_bow = bow_stem[:1464,:]
      test_bow = bow_stem[1464,:]

      xtrain_bow, xvalid_bow, ytrain, yvalid = train_test_split(train_bow, df_train['airline_sentiment'], random_state=42, test_size=0.3)

      lreg = LogisticRegression()
      lreg.fit(xtrain_bow, ytrain)
      prediction = lreg.predict_proba(xvalid_bow)
      prediction_int = prediction[:,1] >= 0.3
      prediction_int = prediction_int.astype(np.int)

      A1 = f1_score(yvalid, prediction_int)
      print(A1)

0.7214611872146119

```

Figure 7
Testing accuracy score for stemmed tweets

```

twittersentimentanalysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

      train_bow = bow_lemm[:1464,:]
      test_bow = bow_lemm[1464,:]

      xtrain_bow, xvalid_bow, ytrain, yvalid = train_test_split(train_bow, df_train['airline_sentiment'], random_state=42, test_size=0.3)

      lreg = LogisticRegression()
      lreg.fit(xtrain_bow, ytrain)

      prediction = lreg.predict_proba(xvalid_bow)
      prediction_int = prediction[:,1] >= 0.3
      prediction_int = prediction_int.astype(np.int)

      A2 = f1_score(yvalid, prediction_int)
      print(A2)

0.7285382830626451

```

Figure 8
Testing accuracy score for lemmatized tweets

To forecast the sentiment of airline tweets, this code is setting up a logistic regression model. The data set is split into a training set and a test set, and the logistic regression model is then fitted using the training set. The F1 score is used to gauge the model's accuracy after it has been evaluated on the test set.

To forecast the tone of airline tweets, this code is building a logistic regression model. The data is first divided into training and test data. It then develops a logistic regression model using the training data. The model is then used to forecast the test data's emotional tone, and lastly, its accuracy is evaluated using an F1 score.

5. Discussion

In the last few years, Python has become more and more popular for data analysis and machine learning. Python is a good choice for data analysis and machine learning jobs due to its simplicity of use, a large library of pre-built functions, and availability of data analysis and "machine learning algorithms". Exploration and extraction are the two components of data analysis in Python. Data is viewed and examined during the exploration phase in order to get insights and spot patterns in the data. Applying a variety of data analysis methods and tools, including "statistical analysis, numerical calculations, graphical displays, data wrangling, and data mining," is how this is accomplished. The process of data extraction is used to draw out relevant information from the data once it has been evaluated and key insights have been recognised. Because it supports simple coding, has a large selection of libraries for "numerical computations," and provides excellent accessibility, Python is also a fantastic language for using "machine learning techniques." Python offers libraries and APIs for many well-known machine-learning techniques, including Scikit-Learn, TensorFlow, and Keras. Together, these tools make Python an effective tool for machine learning applications.

6. Conclusion

Tabular Data set:

After analysing each model, we can draw the following conclusions:

Accuracy: When it comes to model accuracy, the Decision Tree outperforms the Random Forest Tree classifier, which outperforms the KNN classifier and Logistic classification in turn.

F1-scores are:

1. Logistic classification Model: (accuracy = 0.48)
2. Decision tree classifier Model: (accuracy = 0.85)
3. Random forest classifier Model: (accuracy = 0.67)
4. KNN classifier Model: (accuracy = 0.54)

We, therefore conclude that the Decision tree classifier model is the best model for the above data set with an accuracy of 0.85

Text Data set:

After analysing each model in the Twitter sentiment data, we can draw the following conclusions:

Accuracy: When it comes to model accuracy, the Decision tree classifier outperforms the other models.

F1-scores are:

1. Random Forest Classifier Model: (accuracy = 0.756)
2. XGB Classifier Model: (accuracy = 0.715)
3. Gradient Boosting Model: (accuracy = 0.720)
4. Decision Tree Model: (accuracy = 0.99)

We, therefore conclude that the Decision tree classifier model is the best model for the above data set with an accuracy of 0.99.