INTELLIGENT DATA ANALYSIS & MACHINE LEARNING

Heena Manglani

Matricualtion Number - 819452

THE TASK

- To create an email classifier which identifies email as a spam or non-spam from total of 57,173 different words (features) are distinguished.
- The aim of the filter is to identify a maximum number of spam emails, with a maximum of 0.2% of all legitimate emails being classified incorrectly. In addition, the company wants to make a statement about the effectiveness of the filter on future emails, i.e., what percentage of incoming spam emails will be identified in the future.

AGENDA

- The data and pre-processing
- Model and metrics selection
- Comparing results and hyperparameter tuning
- Discussing the objectives
- Conclusions

UNDERSTANDING DATA

- How big is the data?
- How does the data look like?
- What is the data type of columns?
- Are there any missing values?
- Are there any duplicates?
- Is there any corelation between columns in dataType(Categorical of Numerical)?

THE DATA

- Total 10,000 emails
- 57173 - different words(features)
- All numerical
- Imbalanced data – more legitimate(not-spam) data
- Fixed, given dataset(In vectors), no need for collection and update protocols
- Pre-processing is not undertaken.

MODEL AND METRIC SELECTION

- Logistic Regression
- Naïve Bayes
- Random Forest Classifier
- Neural Network
- Accuracy and confusion matrix

SOURCE - HTTPS://MEDIUM.COM/HUGO-FERREIRAS-BLOG/CONFUSION-MATRIX-AND-OTHER-METRICS-IN-MACHINE-LEARNING-894688CB1C0A

Logistic Regression(Linear Classification)

```
# 1. Train the model
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train,y_train)



# 2. Prediction
y_pred = model.predict(X_test)



# 3. Evaluate
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
false_positive_rate = fp / (fp + tn)
true_positive_rate = tp / (tp + fn)

print("Accuracy of the model is : ", round(accuracy_score(y_test, y_pred)*100,2),"%\n\n")
print("Confusion Matrix : \n", confusion_matrix(y_test, y_pred))
print("\n True Positive Rate : ",true_positive_rate )
print("\n False Positive Rate : ",false_positive_rate )
```

```
Accuracy of the model is :  99.65 %


Confusion Matrix :
 [[ 368    5]
 [   2 1625]]

 True Positive Rate :  0.998770743700615

 False Positive Rate :  0.013404825737265416
```

```python
from sklearn.model_selection import cross_val_score

print("\n---- Cross validation on actual dataset---\n ")
score = cross_val_score(model, X, y, cv = 10)
print(score)
print("\n Average :",round(score.mean() *100, 2))

# Cross validation on balanced dataset
print("\n---- Cross validation on balanced dataset---\n ")
score = cross_val_score(model, X_balanced, y_balanced, cv = 10)
print(score)
print("\n Average :",round(score.mean() *100, 2))
```

```
---- Cross validation on actual dataset---

[0.986 0.994 0.993 0.996 0.998 0.997 0.994 0.999 0.992 0.998]

 Average : 99.47

---- Cross validation on balanced dataset---

[0.99252802 0.99564134 0.998132   0.99875467 0.99937733 1.
 1.         1.         0.99750934 0.998132  ]

 Average : 99.8
```

```python
from sklearn.naive_bayes import MultinomialNB

# 1. Model build and train
model = MultinomialNB()              # create a model
model.fit(X_train,y_train)           # train the model

# 2. Predict
y_pred = model.predict(X_test)       # test the model

# 3. Evaluate
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
false_positive_rate = fp / (fp + tn)
true_positive_rate = tp / (tp + fn)

print("Accuracy of the model is : ", round(accuracy_score(y_test, y_pred)*100,2),"%\n\n")
print("Confusion Matrix : \n", confusion_matrix(y_test, y_pred))
print("\n True Positive Rate : ",true_positive_rate )
print("\n False Positive Rate : ",false_positive_rate )
```

```
Accuracy of the model is :  97.9 %


Confusion Matrix :
 [[ 372    1]
 [  41 1586]]

 True Positive Rate :  0.97480024585126

 False Positive Rate :  0.002680965147453083
```

CROSS VALIDATION – NAÏVE BAYES

```
params = { "alpha" : [0, 0.5 , 1.0] }  # s

# Hyper parameter tuning
random_search = RandomizedSearchCV(model, 
                                   n_jobs 
random_search.fit(X,y)

random_search.best_estimator_
```

```
Fitting 5 folds for each of 3 candi
[CV 1/5] END ......................
[CV 2/5] END ......................
[CV 3/5] END ......................
[CV 4/5] END ......................
[CV 5/5] END ......................
[CV 1/5] END ......................
[CV 2/5] END ......................
[CV 3/5] END ......................
[CV 4/5] END ......................
[CV 5/5] END ......................
[CV 1/5] END ......................
[CV 2/5] END ......................
[CV 3/5] END ......................
[CV 4/5] END ......................
[CV 5/5] END ......................
▾ MultinomialNB
MultinomialNB(alpha=0)
```

NEURAL NETWORK MODEL

ACCURACY – 70.4%

```
#fit ANN to training set
model_history = model.fit(X_train.toarray(), y_train, validation_split = 0.30,

print(model_history.history.keys())


# 2. Predict
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)


# 3. Evaluate
print("Accuracy of the model is : ", round(accuracy_score(y_test, y_pred)*100,
```

```
Epoch 1/10
6/6 [==============================] - 20s 3s/step - loss: -2.14
Epoch 2/10
6/6 [==============================] - 17s 3s/step - loss: -40.5
Epoch 3/10
6/6 [==============================] - 17s 3s/step - loss: -233.
Epoch 4/10
6/6 [==============================] - 17s 3s/step - loss: -983.
Epoch 5/10
6/6 [==============================] - 17s 3s/step - loss: -3122
Epoch 6/10
6/6 [==============================] - 17s 3s/step - loss: -8199
Epoch 7/10
```

RANDOM-FOREST CLASSIFIER

```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

```
Accuracy of the model is :  99.65 %


Confusion Matrix :
 [[ 368    5]
```

CONCLUSION

- Logistic Regression and Naive Bayes on balanced dataset works the best for the given dataset.

- What percentage of incoming spam emails will be identifed in the future? - Logistic regression model assure that 99.88% accuracy of the incoming spam emails will be identified in the future.


THANK YOU