Week_5

July 9, 2023

[1]:
```
import pandas as pd
```

[2]:
```
# Creates first dataset from TSV file
data=pd.read_csv('labeledTrainData.tsv',sep='\t')
data['review'] = data['review'].apply(str)
data
```

[2]:

	id	sentiment	review
0	5814_8	1	With all this stuff going down at the moment w…
1	2381_9	1	\The Classic War of the Worlds\" by Timothy Hi…
2	7759_3	0	The film starts with a manager (Nicholas Bell)…
3	3630_4	0	It must be assumed that those who praised this…
4	9495_8	1	Superbly trashy and wondrously unpretentious 8…
…	…	…	…
24995	3453_3	0	It seems like more consideration has gone into…
24996	5064_1	0	I don't believe they made this film. Completel…
24997	10905_3	0	Guy is a loser. Can't get girls, needs to buil…
24998	10194_3	0	This 30 minute documentary Buñuel made in the …
24999	8478_8	1	I saw this movie as a child and it broke my he…

[25000 rows x 3 columns]

[3]:
```
from nltk.stem.porter import PorterStemmer
```

[4]:
```
# Creates porter
porter = PorterStemmer()
```

[5]:
```
# Creates decapitilzer for strings
def decapitalizer(string: str) -> str:
    return string.lower()
```

[6]:
```
# Applies decapitilizer
data['review'] = data['review'].apply(decapitalizer)
```

[7]:
```
# Splits review into tokens
data['review'] = data['review'].str.split()
data
```

[7]:

	id	sentiment	review
0	5814_8	1	[with, all, this, stuff, going, down, at, the,…
1	2381_9	1	[\the, classic, war, of, the, worlds\", by, ti…
2	7759_3	0	[the, film, starts, with, a, manager, (nichola…
3	3630_4	0	[it, must, be, assumed, that, those, who, prai…
4	9495_8	1	[superbly, trashy, and, wondrously, unpretenti…
…	…	…	…
24995	3453_3	0	[it, seems, like, more, consideration, has, go…
24996	5064_1	0	[i, don't, believe, they, made, this, film., c…
24997	10905_3	0	[guy, is, a, loser., can't, get, girls,, needs…
24998	10194_3	0	[this, 30, minute, documentary, buñuel, made, …
24999	8478_8	1	[i, saw, this, movie, as, a, child, and, it, b…

[25000 rows x 3 columns]

[8]:
```
# Applies stem to column review
data['review'] = data['review'].apply(lambda x: [porter.stem(word) for word in␣
 ↪x])
data
```

[8]:

	id	sentiment	review
0	5814_8	1	[with, all, thi, stuff, go, down, at, the, mom…
1	2381_9	1	[\the, classic, war, of, the, worlds\", by, ti…
2	7759_3	0	[the, film, start, with, a, manag, (nichola, b…
3	3630_4	0	[it, must, be, assum, that, those, who, prais,…
4	9495_8	1	[superbl, trashi, and, wondrous, unpretenti, 8…
…	…	…	…
24995	3453_3	0	[it, seem, like, more, consider, ha, gone, int…
24996	5064_1	0	[i, don't, believ, they, made, thi, film., com…
24997	10905_3	0	[guy, is, a, loser., can't, get, girls,, need,…
24998	10194_3	0	[thi, 30, minut, documentari, buñuel, made, in…
24999	8478_8	1	[i, saw, thi, movi, as, a, child, and, it, bro…

[25000 rows x 3 columns]

[18]:
```
# Re-joins all the tokens to one string
data["review"]= data["review"].str.join(" ")
data
```

[18]:

	id	sentiment	review
0	5814_8	1	with all thi stuff go down at the moment with …
1	2381_9	1	\the classic war of the worlds\" by timothi hi…
2	7759_3	0	the film start with a manag (nichola bell) giv…
3	3630_4	0	it must be assum that those who prais thi film…
4	9495_8	1	superbl trashi and wondrous unpretenti 80' exp…
…	…	…	…
24995	3453_3	0	it seem like more consider ha gone into the im…

```
24996   5064_1          0  i don't believ they made thi film. complet unn…
24997  10905_3          0  guy is a loser. can't get girls, need to build…
24998  10194_3          0  thi 30 minut documentari buñuel made in the ea…
24999   8478_8          1  i saw thi movi as a child and it broke my hear…

[25000 rows x 3 columns]
```

Split this into a training and test set.

[19]:
```
from sklearn.model_selection import train_test_split
```

[74]:
```
# Separate the target from the features
feature = data['review']
target = data['sentiment']

#Split the data into 80% training and 20% test
feature_train, feature_test, target_train, target_test =␣
 ↪train_test_split(feature, target, test_size=0.20, random_state=42)
```

Fit and apply the tf-idf vectorization to the training set.

[21]:
```
# Import libraries
from sklearn.feature_extraction.text import TfidfVectorizer
```

[22]:
```
# Creates Tfid Vectorizer
tfidf = TfidfVectorizer()
```

[23]:
```
# Creates feature matrix
feature_matrix = tfidf.fit_transform(feature_train)
```

Apply but DO NOT FIT the tf-idf vectorization to the test set (Why?)

[25]:
```
test_feature_matrix = tfidf.transform(feature_test)
```

Train a logistic regression using the training data.

[26]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

[27]:
```
logistical_regression = LogisticRegression()
```

[29]:
```
# Fit the model to the training data
model = logistical_regression.fit(feature_matrix, target_train)
```

Find the model accuracy on the test set.

[30]:
```
from sklearn import metrics
```

[34]:
```
# Create predictions
prediction = logistical_regression.predict(feature_matrix)
# Calculate the accuracy
accuracy = 100*metrics.accuracy_score(prediction,target_train)
# Display accuracy
print('The accuracy of the Logistic Regression is: ', round(accuracy,2), '%',␣
 ↪sep = '')
```

```
The accuracy of the Logistic Regression is: 92.62%
```

Create a confusion matrix for the test set predictions

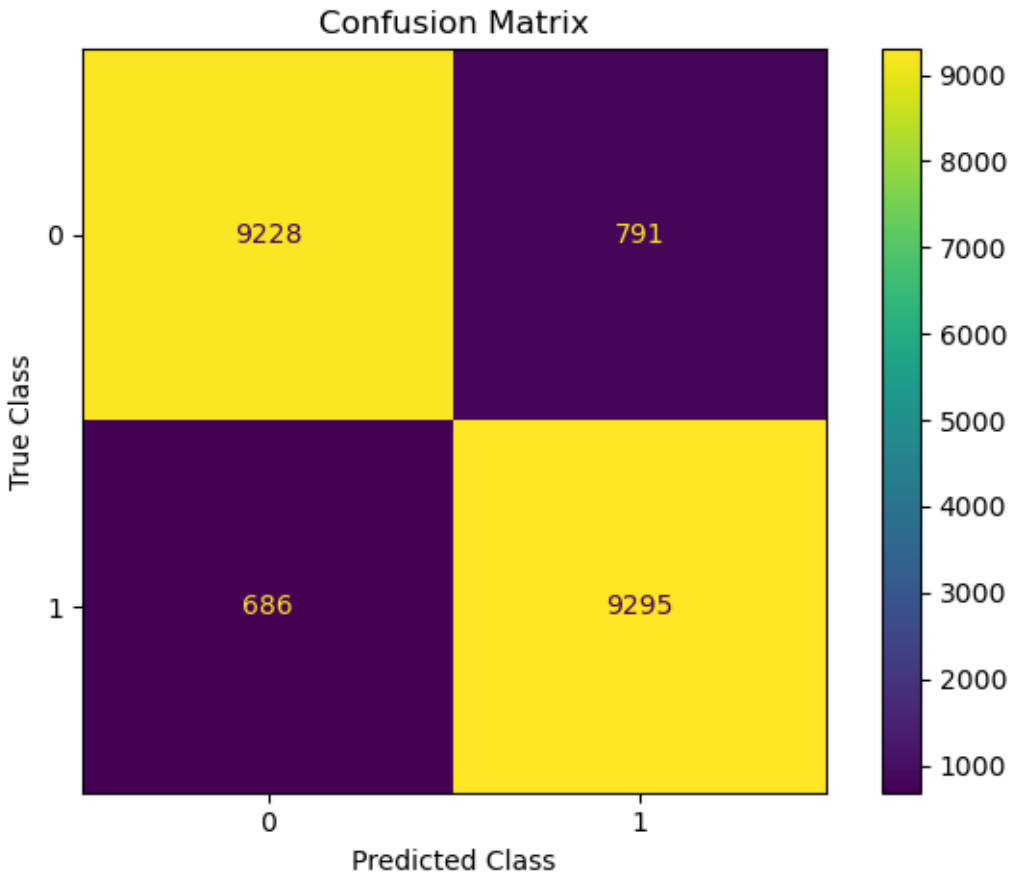
[132]:
```
# Creates predictions
target_predicted = logistical_regression.fit(feature_matrix, target_train).
 ↪predict(feature_matrix)
```

[82]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

[83]:
```
# Creates confusion matric using train set and predictions
c_matrix = confusion_matrix(target_train, target_predicted)
```

[84]:
```
# Configures confusion matrix to be able to display
disp = ConfusionMatrixDisplay(confusion_matrix=c_matrix)
```

[85]:
```
# Creates Plot
disp.plot()
# Creates title
plt.title("Confusion Matrix"), plt.tight_layout()
# Creates y and x labels
plt.ylabel("True Class"), plt.xlabel("Predicted Class")
# Displays plot
plt.show()
```



Get the precision, recall, and F1-score for the test set predictions.

[77]:
```
from sklearn.metrics import classification_report
```

[87]:
```
# Displays precision, recall and F-1 Score
print(classification_report(target_train, target_predicted))
```

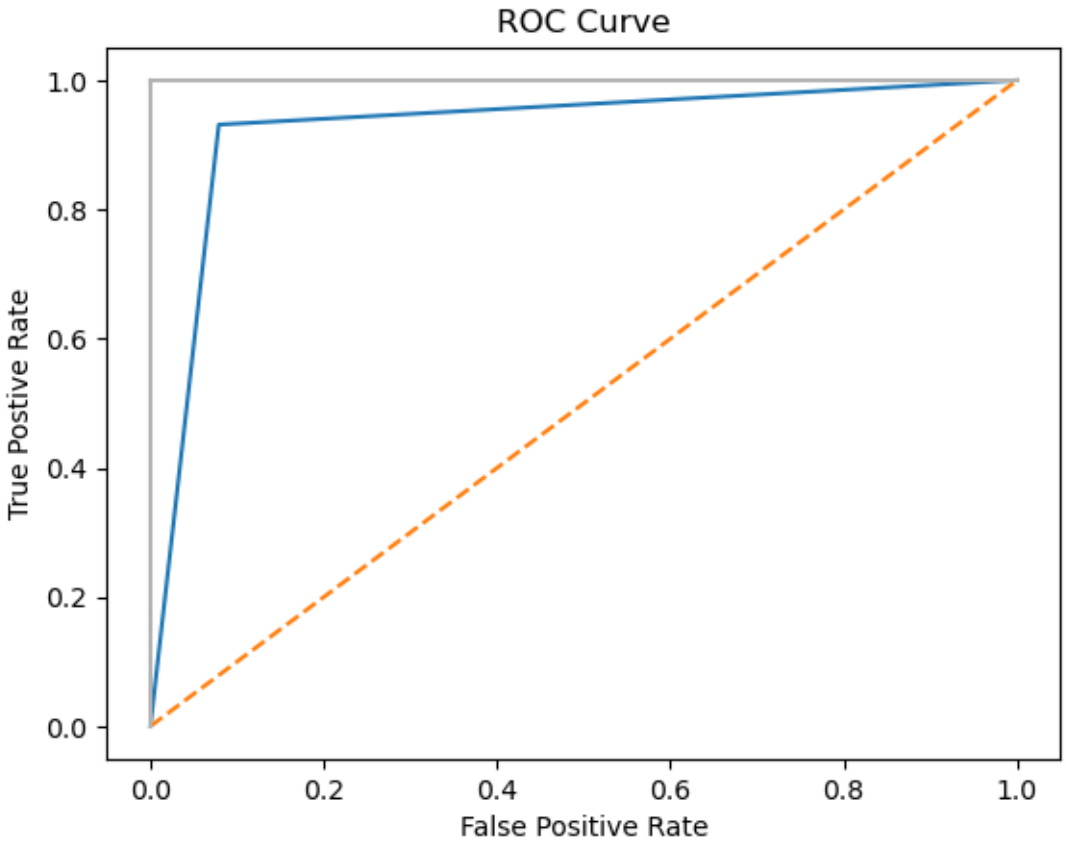
```
              precision    recall  f1-score   support

           0       0.93      0.92      0.93     10019
           1       0.92      0.93      0.93      9981

    accuracy                           0.93     20000
   macro avg       0.93      0.93      0.93     20000
weighted avg       0.93      0.93      0.93     20000
```

Create a ROC curve for the test set.

[90]:
```
from sklearn.metrics import roc_curve, roc_auc_score
```

[103]:
```
# Creates rates for true and false
false_positive_rate, true_positive_rate, threshold = roc_curve(target_train,␣
↪target_predicted)
```

[110]:
```
# Title for ROC Curve
plt.title("ROC Curve")
# Creates plot using false and true
plt.plot(false_positive_rate, true_positive_rate)
# Creates center line
plt.plot([0,1], ls="--")
plt.plot([0, 0], [1,0], c=".7"), plt.plot([1,1], c=".7")
# Creates Labels
plt.xlabel("False Positive Rate")
plt.ylabel("True Postive Rate")
# Displays plot
plt.show()
```



Random Forest Model

[112]:
```
from sklearn.tree import DecisionTreeClassifier
```

[113]:
```
# Creates decision tree classifier function
decisiontree = DecisionTreeClassifier()
```

Train a classification regression model using the training data

[115]:
```
# Trains a model using training data
model_tree = decisiontree.fit(test_feature_matrix, target_test)
```

Find the model accuracy on the test set.

[118]:
```
# Creates predictions
prediction = decisiontree.predict(feature_matrix)
# Calculates accuracy
accuracy = 100*metrics.accuracy_score(prediction,target_train)
# Displays Accuracy
print('The accuracy of the Random Forest Model is: ', round(accuracy,2), '%',␣
 ↪sep = '')
```

```
The accuracy of the Random Forest Model is: 67.92%
```

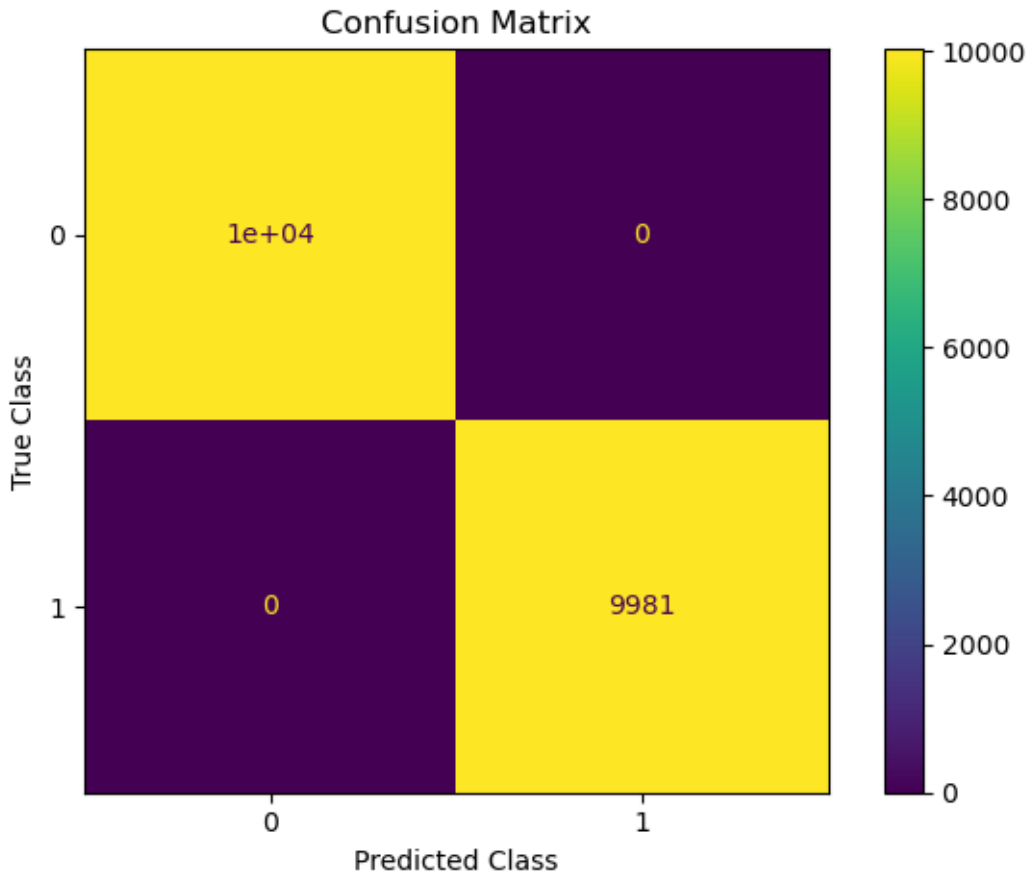
Create a confusion matrix for the test set predictions.

[120]:
```
# Creates predictions using decision tree
target_predicted_tree = decisiontree.fit(feature_matrix, target_train).
 ↪predict(feature_matrix)
```

[124]:
```
# Creates confusion matrix using predictions from decision tree
c_matrix_tree = confusion_matrix(target_train, target_predicted_tree)
```

[126]:
```
# Configures confusion matrix to be able to display
disp_tree = ConfusionMatrixDisplay(confusion_matrix=c_matrix_tree)
```

[127]:
```
# Creates plot
disp_tree.plot()
# Creates titles
plt.title("Confusion Matrix"), plt.tight_layout()
# Creates labels
plt.ylabel("True Class"), plt.xlabel("Predicted Class")
# Displays plot
plt.show()
```



Get the precision, recall, and F1-score for the test set predictions.

[128]:
```
# Displays precision, recall and F-1 Score
print(classification_report(target_train, target_predicted_tree))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     10019
           1       1.00      1.00      1.00      9981

    accuracy                           1.00     20000
   macro avg       1.00      1.00      1.00     20000
weighted avg       1.00      1.00      1.00     20000
```

Create a ROC curve for the test set.

[129]:
```
# Creates rates for true and false
false_positive_rate_tree, true_positive_rate_tree, threshold_tree =␣
 ↪roc_curve(target_train, target_predicted_tree)
```

[130]:
```
# Creates title
plt.title("ROC Curve")
# Creates plot using false and true
plt.plot(false_positive_rate_tree, true_positive_rate_tree)
# Creates center line
plt.plot([0,1], ls="--")
plt.plot([0, 0], [1,0], c=".7"), plt.plot([1,1], c=".7")
# Creates labels
plt.xlabel("False Positive Rate")
plt.ylabel("True Postive Rate")
# Displays plot
plt.show()
```