ImplementMLProjectPlan

August 12, 2023

1 Lab 8: Implement Your Machine Learning Project Plan

In this lab assignment, you will implement the machine learning project plan you created in the written assignment. You will:

1. Load your data set and save it to a Pandas DataFrame.
2. Perform exploratory data analysis on your data to determine which feature engineering and data preparation techniques you will use.
3. Prepare your data for your model and create features and a label.
4. Fit your model to the training data and evaluate your model.
5. Improve your model by performing model selection and/or feature selection techniques to find best model for your problem.

1.0.1 Import Packages

Before you get started, import a few packages.

[2]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

Task: In the code cell below, import additional packages that you have used in this course that you will need for this task.

[16]:
```
# YOUR CODE HERE
from sklearn.metrics import plot_confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪confusion_matrix
import string
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
```

1.1 Part 1: Load the Data Set

You have chosen to work with one of four data sets. The data sets are located in a folder named "data." The file names of the three data sets are as follows:

- The "adult" data set that contains Census information from 1994 is located in file `adultData.csv`
- The airbnb NYC "listings" data set is located in file `airbnbListingsData.csv`
- The World Happiness Report (WHR) data set is located in file `WHR2018Chapter2OnlineData.csv`
- The book review data set is located in file `bookReviewsData.csv`

Task: In the code cell below, use the same method you have been using to load your data using `pd.read_csv()` and save it to DataFrame `df`.

[4]:
```
# YOUR CODE HERE
filename = os.path.join(os.getcwd(), "data", "bookReviewsData.csv")
df = pd.read_csv(filename, header=0)
```

[5]:
```
df.head()
```

[5]:

	Review	Positive Review
0	This was perhaps the best of Johannes Steinhof...	True
1	This very fascinating book is a story written ...	True
2	The four tales in this collection are beautifu...	True
3	The book contained more profanity than I expec...	False
4	We have now entered a second time of deep conc...	True

1.2 Part 2: Exploratory Data Analysis

The next step is to inspect and analyze your data set with your machine learning problem and project plan in mind.

This step will help you determine data preparation and feature engineering techniques you will need to apply to your data to build a balanced modeling data set for your problem and model. These data preparation techniques may include: * addressing missingness, such as replacing missing values with means * renaming features and labels * finding and replacing outliers * performing winsorization if needed * performing one-hot encoding on categorical features * performing vectorization for an NLP problem * addressing class imbalance in your data sample to promote fair AI

Think of the different techniques you have used to inspect and analyze your data in this course. These include using Pandas to apply data filters, using the Pandas `describe()` method to get insight into key statistics for each column, using the Pandas `dtypes` property to inspect the data type of each column, and using Matplotlib and Seaborn to detect outliers and visualize relationships between features and labels. If you are working on a classification problem, use techniques you have learned to determine if there is class imbalance.

Task: Use the techniques you have learned in this course to inspect and analyze your data.

Note: You can add code cells if needed by going to the Insert menu and clicking on Insert Cell Below in the drop-drown menu.

[6]:
```
# YOUR CODE HERE
# Check for missing values
```

```
count = df.isnull().sum()
percentage = ((df.isnull().sum()/len(df)*100))
missing_data = pd.concat([count, percentage], axis=1, keys=['Count',␣
 ↪'Percentage'])

missing_data
```

[6]:
```
                 Count  Percentage
Review               0         0.0
Positive Review      0         0.0
```

[7]:
```
# Renaming features
df.rename(columns={'Review': 'review_text', 'Positive Review':␣
 ↪'label_positive'}, inplace=True)
df.head()
```

[7]:
```
                                         review_text  label_positive
0  This was perhaps the best of Johannes Steinhof...            True
1  This very fascinating book is a story written ...            True
2  The four tales in this collection are beautifu...            True
3  The book contained more profanity than I expec...           False
4  We have now entered a second time of deep conc...            True
```

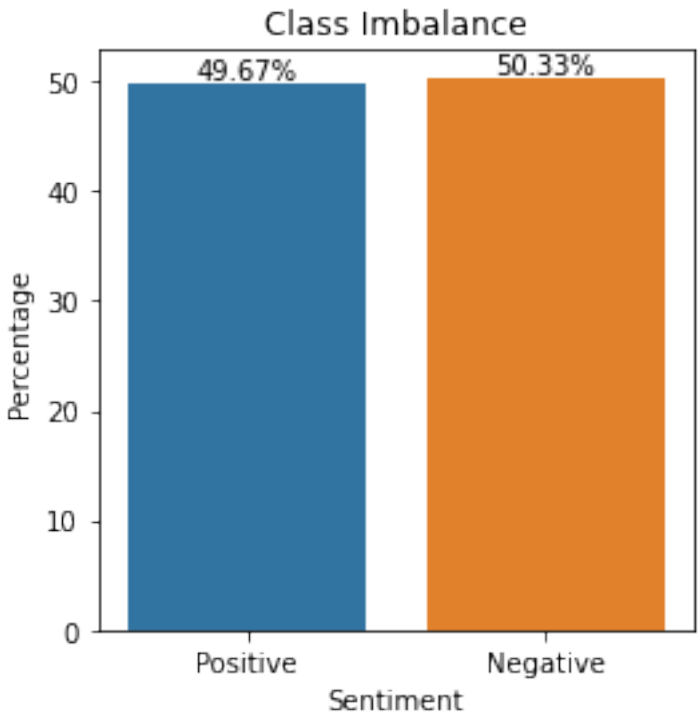
[8]:
```
# Summary statistics of the dataset
summary_stats = df.describe().transpose()
summary_stats
```

[8]:
```
               count unique  \
review_text     1973   1865
label_positive  1973      2

                                                          top freq
review_text     I have read several of Hiaasen's books and lov...    3
label_positive                                              False  993
```

[9]:
```
# Check class imbalance
pos_percentage = df["label_positive"].value_counts(normalize=True)[1]*100
neg_percentage = df["label_positive"].value_counts(normalize=True)[0]*100
print(f'Positive Reviews: {"{:.2f}".format(pos_percentage)}%')
print(f'Negative Reviews: {"{:.2f}".format(neg_percentage)}%')

# Visualize class imbalance
plt.figure(figsize=(4, 4))
sns.barplot(x=["Positive", "Negative"], y=[pos_percentage, neg_percentage])
plt.title("Class Imbalance")
plt.xlabel("Sentiment")
plt.ylabel("Percentage")

# Display percentage values
for index, value in enumerate([pos_percentage, neg_percentage]):
```

```
    plt.text(index, value, f'{value:.2f}%', ha='center', va='bottom')


plt.show()
```

```
Positive Reviews: 49.67%
Negative Reviews: 50.33%
```



[10]:
```
# Cleaning reviews content
def cleaning(text):
    # Lowercase
    text = text.lower()
    # Tokenize the text into words using spaces
    words = text.split()
    # Remove punctuation from each word
    text = [word.strip(string.punctuation) for word in words]
    # Remove digits
    text = ["".join(c for c in word if not c.isdigit()) for word in text]
    text = " ".join(text)

    return text

# Cleaning text in dataset
cleaned = lambda x: cleaning(x)
```

```
df['cleaned_review_text'] = pd.DataFrame(df['review_text'].apply(cleaned))
df.head()
```

[10]:
```
                                         review_text  label_positive  \
0  This was perhaps the best of Johannes Steinhof...            True
1  This very fascinating book is a story written ...            True
2  The four tales in this collection are beautifu...            True
3  The book contained more profanity than I expec...           False
4  We have now entered a second time of deep conc...            True

                                cleaned_review_text
0  this was perhaps the best of johannes steinhof...
1  this very fascinating book is a story written ...
2  the four tales in this collection are beautifu...
3  the book contained more profanity than i expec...
4  we have now entered a second time of deep conc...
```

1.3 Part 3: Implement Your Project Plan

Task: Use the rest of this notebook to carry out your project plan. You will:

1. Prepare your data for your model and create features and a label.
2. Fit your model to the training data and evaluate your model.
3. Improve your model by performing model selection and/or feature selection techniques to find best model for your problem.

Add code cells below and populate the notebook with commentary, code, analyses, results, and figures as you see fit.

[11]:
```
# YOUR CODE HERE
# Create labels
y = df['label_positive']
X = df['cleaned_review_text']

X.head()
```

[11]:
```
0    this was perhaps the best of johannes steinhof...
1    this very fascinating book is a story written ...
2    the four tales in this collection are beautifu...
3    the book contained more profanity than i expec...
4    we have now entered a second time of deep conc...
Name: cleaned_review_text, dtype: object
```

[12]:
```
# Splitting labeled into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,␣
 ↪random_state=1234)
```

[13]:
```
X_train.head()
```

[13]:
```
1093    sorry this guy's voice gave me the creeps he s...
485     this novel could not hold my interest i am an ...
```

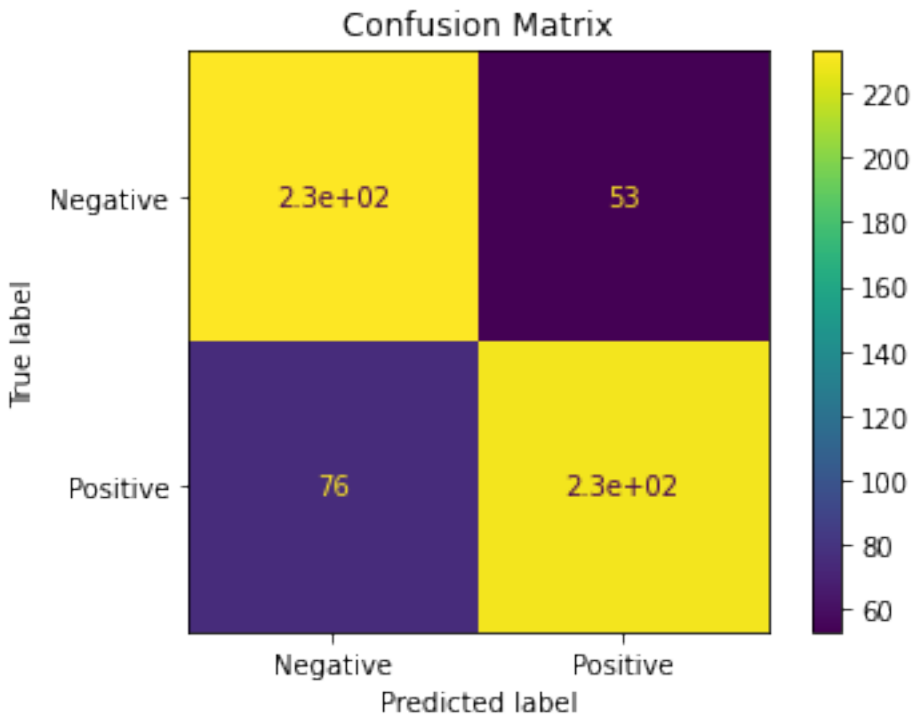
```
390     the photo presented here is a bit too bright a...
60      the problem with this book is that it does not...
435     the author gave many thorough and enlightening...
Name: cleaned_review_text, dtype: object
```

[14]:
```
# Try confusion matrix
# Create a TfidVectorizer object
tfidf_vect = TfidfVectorizer()
# Initialize the model
model_rf = RandomForestClassifier(criterion='entropy', n_estimators=100)

model = Pipeline([('vectorizer', tfidf_vect), ('classifier', model_rf)])

# Fit the vectorizer to X_train
# Fit the model
model.fit(X_train, y_train)

predictions = model.predict(X_test)

confusion_matrix(predictions, y_test)
```

[14]:
```
array([[233,  76],
       [ 53, 230]])
```

[18]:
```
plt.figure(figsize=(8, 6))
plot_confusion_matrix(model, X_test, y_test, display_labels=["Negative",␣
 ↪"Positive"])
plt.title("Confusion Matrix")
plt.show()
```

```
<Figure size 576x432 with 0 Axes>
```



[19]:
```
accuracy = accuracy_score(predictions, y_test)
precision = precision_score(predictions, y_test)
recall = recall_score(predictions, y_test)
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
```

```
Accuracy: 0.7820945945945946
Precision: 0.7516339869281046
Recall: 0.8127208480565371
```

This output shows that the model has been trained kind of well due to high accuracy score, precision, and recall score.

[20]:
```
predictions
```

[20]:
```
array([ True,  True,  True,  True,  True,  True, False, False,  True,
       False, False, False,  True,  True, False,  True, False, False,
        True, False, False, False, False,  True, False, False,  True,
        True, False, False, False, False,  True,  True,  True, False,
       False, False,  True,  True,  True,  True, False, False, False,
       False, False,  True,  True,  True, False, False, False, False,
       False, False,  True,  True, False, False,  True,  True, False,
       False, False,  True, False, False, False, False, False, False,
       False, False,  True, False, False,  True, False, False, False,
```

```
 True, False,  True, False, False, False, False, False, False,
False,  True, False,  True, False, False, False,  True,  True,
False,  True, False,  True, False,  True,  True,  True,  True,
False,  True,  True, False,  True,  True, False, False,  True,
 True,  True,  True, False,  True, False, False,  True, False,
False,  True,  True,  True, False, False,  True,  True, False,
 True,  True,  True,  True, False, False,  True,  True,  True,
 True,  True,  True,  True, False, False,  True, False, False,
 True,  True,  True, False,  True,  True, False,  True,  True,
False, False, False,  True,  True, False, False,  True,  True,
False, False, False,  True, False,  True, False, False,  True,
 True, False, False, False,  True, False, False, False,  True,
False,  True,  True,  True, False,  True,  True,  True,  True,
False, False, False, False,  True,  True, False,  True,  True,
 True,  True,  True, False,  True,  True, False, False,  True,
False,  True,  True, False, False,  True,  True,  True,  True,
 True, False,  True,  True,  True,  True, False,  True, False,
 True, False, False,  True, False,  True, False, False,  True,
 True,  True,  True, False, False, False, False,  True,  True,
False, False,  True, False, False,  True,  True, False,  True,
 True, False, False, False,  True,  True,  True, False,  True,
False, False,  True, False, False,  True,  True,  True, False,
False, False, False,  True, False, False, False, False, False,
 True, False, False,  True,  True,  True, False,  True, False,
 True,  True,  True,  True, False,  True, False, False, False,
False, False,  True, False, False, False, False,  True,  True,
 True, False,  True,  True, False, False,  True, False, False,
 True, False,  True, False, False, False, False,  True,  True,
 True, False,  True,  True,  True, False,  True,  True, False,
 True, False, False, False,  True,  True, False,  True, False,
 True, False, False,  True, False,  True, False,  True,  True,
False, False, False, False, False,  True,  True, False, False,
False,  True, False, False,  True,  True, False, False, False,
False,  True,  True, False, False, False,  True, False,  True,
 True, False, False,  True,  True, False,  True,  True, False,
 True, False,  True,  True,  True, False, False, False,  True,
 True, False,  True,  True, False,  True,  True,  True,  True,
 True, False, False,  True, False, False,  True, False,  True,
 True,  True,  True, False, False, False, False,  True, False,
 True,  True,  True, False, False,  True,  True,  True, False,
 True, False,  True,  True,  True,  True,  True,  True, False,
 True,  True,  True, False, False,  True, False,  True, False,
 True,  True,  True, False, False, False,  True, False, False,
False, False, False, False, False, False, False,  True, False,
False, False, False, False, False,  True,  True, False, False,
False,  True,  True, False, False, False,  True,  True, False,
 True, False, False,  True,  True, False, False, False,  True,
```

```
 True, False, False,  True,  True,  True,  True,  True, False,
False, False,  True,  True, False,  True, False,  True, False,
 True, False, False,  True,  True, False, False,  True,  True,
False,  True, False, False, False, False,  True,  True,  True,
False, False,  True,  True, False, False,  True,  True,  True,
False,  True,  True, False, False, False, False,  True, False,
False, False, False, False, False, False,  True,  True,  True,
 True,  True,  True,  True, False,  True,  True, False, False,
False,  True, False, False, False,  True, False, False,  True,
 True, False, False, False,  True, False, False])
```

[21]:
```
y_test
```

[21]:
```
1692    False
1744     True
1236     True
21      False
894      True
        ...
634     False
1347    False
147      True
1312     True
227     False
Name: label_positive, Length: 592, dtype: bool
```

[22]:
```
# Evaluate model performance
accuracy = accuracy_score(y_test, predictions)
print('Accuracy score: ', accuracy)
```

```
Accuracy score:  0.7820945945945946
```

[23]:
```
# Try to improving model performance by cross-validation
accuracy_cvs = cross_val_score(model, X_train, y_train, cv=5,␣
 ↪scoring='accuracy')
# Print mean and standard deviation of accuracy scores
print('Accuracy Cross-Validation: ', accuracy_cvs)
print('Mean Accuracy Score: ', accuracy_cvs.mean())
print('The standard deviation: ', accuracy_cvs.std())
```

```
Accuracy Cross-Validation:  [0.77617329 0.78985507 0.75724638 0.76086957
0.80434783]
Mean Accuracy Score:  0.7776984251556531
The standard deviation:  0.017673695165127987
```

This process doesn't help improve performance at all.

[24]:
```
print('Grid Search CV')
# Grid search CV
```

```
# Define the hyperparameter grid to search
param_grid = {
    'classifier__n_estimators': [100, 200, 300],
    'classifier__max_depth': [10, 20],
    'classifier__min_samples_split': [2, 4]
}

# Create GridSearchCV object with pipeline
grid_search = GridSearchCV(estimator=model, param_grid=param_grid,␣
 ↪scoring='accuracy', cv=5)
# Fit GridSearchCV
grid_search.fit(X_train, y_train)
# Get the best pipeline
best_pipeline = grid_search.best_estimator_
# Evaluate best pipeline
acc = best_pipeline.score(X_test, y_test)

print('Best Test Accuracy: ', acc)
```

```
Grid Search CV
Best Test Accuracy:  0.8091216216216216
```

By using Grid search CV, the model is better trained with high accuracy 0.8091

[25]:
```
best_params = grid_search.best_params_
print('Best Hyperparameters: ', best_params)
```

```
Best Hyperparameters:  {'classifier__max_depth': 20,
'classifier__min_samples_split': 4, 'classifier__n_estimators': 300}
```

[ ]: