Project name: Data Analyst:: Data Collection Pipeline (Data Acquistion to Storytelling) - Data Collection Pipeline (Data Acquisition to Story telling)

Debika Mukherjee (Solo project)
Email: debika.isms@gmail.com
Country: United Kingdom
University: Cardiff metropolitan university
Specialization: Data Analyst

Problem description:
XYZ company is collecting the data customer using google forms/survey monkey and they have floated n number of forms on the web.
Company wants to create a pipeline which will collect all the data of these google forms/survey monkey and visualize the data in the dashboard.
Company wants clean data and if there is any data issue present in the data then it should be treated by this pipeline (duplicate data or junk data). dedup check should be performed on the email id of the customer

Github repository:
debikaisms/Final-project: Data Collection Pipeline (Data Acquistion to Storytelling) Data Collection Pipeline (Data Acquisition to Story telling) (github.com)

Model Selection-

I performed 4 models hear to understand the best suitable model and their performance on the data sets.

1) Decision tree model -

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset (replace 'your_dataset.csv' with the actual file path)
data = pd.read_excel('surveydata.xlsx')
```

```
# Split the data into features (X) and the target variable (y)
X = data[['Age', 'Satisfaction Rating (1-5)']]
y = data['Product Feedback']
```

```
# Split the data into a training set and a testing set (adjust the test_size as needed)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Create a Decision Tree classifier
clf = DecisionTreeClassifier(random_state=42)
```

```
# Train the classifier on the training data
clf.fit(X_train, y_train)
```

```
# Make predictions on the test data
y_pred = clf.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

```
# Print the results
print(f'Accuracy: {accuracy}')
```

```
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

Result - Accuracy: 0.5
Confusion Matrix:
[[0 1 0]
 [0 0 0]
 [0 0 1]]
Classification Report:

	precision	recall	f1-score	support
Excellent service	0.00	0.00	0.00	1
Good customer support	0.00	0.00	0.00	0
Great product, very satisfied	1.00	1.00	1.00	1
accuracy			0.50	2
macro avg	0.33	0.33	0.33	2
weighted avg	0.50	0.50	0.50	2

Conclusion - The conclusion of the Decision Tree classification results is as follows:

Accuracy: The accuracy of the model is 50%, which means it correctly predicted the class for 50% of the test samples.

Confusion Matrix:

For the class "Excellent service," the model correctly predicted 0 samples and incorrectly predicted 1 sample.
For the class "Good customer support," there were no correct predictions (0 samples).
For the class "Great product, very satisfied," the model correctly predicted 1 sample.
Classification Report:

The precision, recall, and F1-score for the class "Excellent service" are all 0. This suggests that the model did not correctly identify any instances of this class.
Since there were no samples for the class "Good customer support" in the test set, precision, recall, and F1-score for this class are also 0.
For the class "Great product, very satisfied," the model achieved perfect precision, recall, and F1-score, indicating that it correctly classified all instances of this class.
Overall:

The model has a macro-average F1-score of 0.33, indicating poor overall performance.
The weighted average F1-score is also 0.50, reflecting the balanced contribution of each class to the metric.

2) Logistic Regression -

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset (replace 'your_dataset.csv' with the actual file path)
data = pd.read_excel('surveydata.xlsx')
```

```
# Split the data into features (X) and the target variable (y)
X = data[['Age', 'Satisfaction Rating (1-5)']]
y = data['Product Feedback']
```

```
# Split the data into a training set and a testing set (adjust the test_size as needed)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Create a Logistic Regression model
clf = LogisticRegression(random_state=42)
```

```
# Train the model on the training data
clf.fit(X_train, y_train)
```

```
# Make predictions on the test data
y_pred = clf.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

```
# Print the results
print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)
print
```

Result - Accuracy: 0.0
Confusion Matrix:
[[0 1 0]
 [0 0 0]
 [0 1 0]]

Conclusion -

Accuracy: 0.0:

The accuracy is a measure of how many predictions made by the model were correct. In this case, the accuracy is 0.0, which means that the model did not correctly predict any samples in the test dataset.
Confusion Matrix:

A confusion matrix is a table used to evaluate the performance of a classification algorithm. It shows the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions.
The confusion matrix you provided:

lua
Copy code
[[0 1 0]
 [0 0 0]
 [0 1 0]]
Breakdown:

The rows of the confusion matrix represent the true classes, and the columns represent the predicted classes.
The class labels are not mentioned, but it seems there are three classes.
Specifically:

The first row (row 1) corresponds to the true class 1 (or the first class). There are no correct predictions for this class (0 TP).

The second row (row 2) corresponds to the true class 2 (or the second class). There are no correct predictions for this class (0 TP).
The third row (row 3) corresponds to the true class 3 (or the third class). There are no correct predictions for this class (0 TP).

3) Support vector machine (SVM)-

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset (replace 'your_dataset.csv' with the actual file path)
data = pd.read_excel('surveydata.xlsx')
```

```
# Split the data into features (X) and the target variable (y)
X = data[['Age', 'Satisfaction Rating (1-5)']]
y = data['Product Feedback']
```

```
# Split the data into a training set and a testing set (adjust the test_size as needed)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Create an SVM classifier (you can choose the kernel type as needed)
clf = SVC(kernel='linear', random_state=42)
```

```
# Train the classifier on the training data
clf.fit(X_train, y_train)
```

```
# Make predictions on the test data
y_pred = clf.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

```
# Print the results
print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

Result -

Accuracy: 0.5
Confusion Matrix:
[[0 1 0]
 [0 0 0]
 [0 0 1]]
Classification Report:

	precision	recall	f1-score	support
Excellent service	0.00	0.00	0.00	1
Good customer support	0.00	0.00	0.00	0
Great product, very satisfied	1.00	1.00	1.00	1
accuracy			0.50	2
macro avg	0.33	0.33	0.33	2
weighted avg	0.50	0.50	0.50	2

Conclusion -

Accuracy: 0.5:

The accuracy is a measure of how many predictions made by the model were correct. In this case, the accuracy is 0.5, which means that the model correctly predicted the class for 50% of the test samples.
Confusion Matrix:

A confusion matrix is a table used to evaluate the performance of a classification algorithm. It shows the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions.
The confusion matrix you provided:

lua
Copy code
[[0 1 0]
 [0 0 0]
 [0 0 1]]
Breakdown:

The rows of the confusion matrix represent the true classes, and the columns represent the predicted classes.
There are three classes, but the class labels are not mentioned.
Specifically:

The first row (row 1) corresponds to the true class 1 (or the first class). There are no correct predictions for this class (0 TP).
The second row (row 2) corresponds to the true class 2 (or the second class). There are no correct predictions for this class (0 TP).
The third row (row 3) corresponds to the true class 3 (or the third class). There is one correct prediction for this class (1 TP).
Classification Report:
The precision, recall, and F1-score are computed for each class based on the confusion matrix.
In this case:

The precision, recall, and F1-score for the class "Excellent service" and "Good customer support" are 0 for both, indicating that the model did not correctly identify any instances of these classes.
For the class "Great product, very satisfied," the model achieved perfect precision, recall, and F1-score, indicating that it correctly classified all instances of this class.
Overall:
The model's accuracy is 50%, but this is primarily due to the perfect performance on one class, "Great product, very satisfied." The model performed poorly on the other classes.

4) Random forest (Ensemble model)-

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset (replace 'your_dataset.csv' with the actual file path)
data = pd.read_excel('surveydata.xlsx')
```

```
# Split the data into features (X) and the target variable (y)
X = data[['Age', 'Satisfaction Rating (1-5)']]
y = data['Product Feedback']
```

```
# Split the data into a training set and a testing set (adjust the test_size as needed)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Create a Random Forest classifier
```

```
clf = RandomForestClassifier(random_state=42)
```

```
# Train the classifier on the training data
clf.fit(X_train, y_train)
```

```
# Make predictions on the test data
y_pred = clf.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

```
# Print the results
print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

Result- Accuracy: 0.5
Confusion Matrix:
[[0 1 0]
 [0 0 0]
 [0 0 1]]
Classification Report:

	precision	recall	f1-score	support
Excellent service	0.00	0.00	0.00	1
Good customer support	0.00	0.00	0.00	0
Great product, very satisfied	1.00	1.00	1.00	1
accuracy			0.50	2
macro avg	0.33	0.33	0.33	2
weighted avg	0.50	0.50	0.50	2

Conclusion -

Accuracy: 0.5:

The accuracy is a measure of how many predictions made by the model were correct. In this case, the accuracy is 0.5, which means that the model correctly predicted the class for 50% of the test samples.
Confusion Matrix:

A confusion matrix is a table used to evaluate the performance of a classification algorithm. It shows the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions.
The confusion matrix you provided:

lua
Copy code
[[0 1 0]
 [0 0 0]
 [0 0 1]]
Breakdown:

The rows of the confusion matrix represent the true classes, and the columns represent the predicted classes.
There are three classes, but the class labels are not mentioned.
Specifically:

The first row (row 1) corresponds to the true class 1 (or the first class). There are no correct predictions for this class (0 TP).
The second row (row 2) corresponds to the true class 2 (or the second class). There are no correct predictions for this class (0 TP).
The third row (row 3) corresponds to the true class 3 (or the third class). There is one correct prediction for this class (1 TP).
Classification Report:
The precision, recall, and F1-score are computed for each class based on the confusion matrix.
In this case:

The precision, recall, and F1-score for the class "Excellent service" and "Good customer support" are 0 for both, indicating that the model did not correctly identify any instances of these classes.
For the class "Great product, very satisfied," the model achieved perfect precision, recall, and F1-score, indicating that it correctly classified all instances of this class.
Overall:
The model's accuracy is 50%, but this is primarily due to the perfect performance on one class, "Great product, very satisfied." The model performed poorly on the other classes.

I performed other models like Adaboost, Nalve bayes, KNN, Nural network, gradient boosting. The details of these code is there in the coding file.

Overall Summary -

Based on the results for various models, it's important to consider the specific evaluation metrics and the context of the problem to determine the best model. Let's summarize the performance of each model:

Decision Tree:

Accuracy: 0.5
Classification results: Suboptimal performance with low accuracy.
Logistic Regression:

Accuracy: 0.0
Classification results: Poor accuracy, the model is not performing well.
Support Vector Machine (SVM):

Accuracy: 0.5
Classification results: Suboptimal performance with low accuracy.
Random Forest (Ensemble Model):

Accuracy: 0.5
Classification results: Suboptimal performance with low accuracy.
AdaBoost (Boosting Model):

Accuracy: 0.5
Classification results: Suboptimal performance with low accuracy.
Naive Bayes (Interpretable):

Accuracy: 0.5
Classification results: Suboptimal performance with low accuracy.
K-Nearest Neighbors (K-NN) (Interpretable):

Accuracy: 0.5
Classification results: Suboptimal performance with low accuracy.

Neural Networks (Deep Learning):

Accuracy: 0.5
Classification results: Suboptimal performance with low accuracy.
Gradient Boosting (XGBoost or LightGBM):

Accuracy: 0.5
Classification results: Suboptimal performance with low accuracy.
Based on these results, none of the models have demonstrated strong predictive performance. The accuracy for all models is low, and the classification results show a mix of false positives and false negatives.