CMSC178DA

Activity 5

Applying Logistic Regression to Binary Classification Problem

Submitted by:

Peladas, Daenielle Rai

Section B (TF 10:30 AM - 12:00 PM)

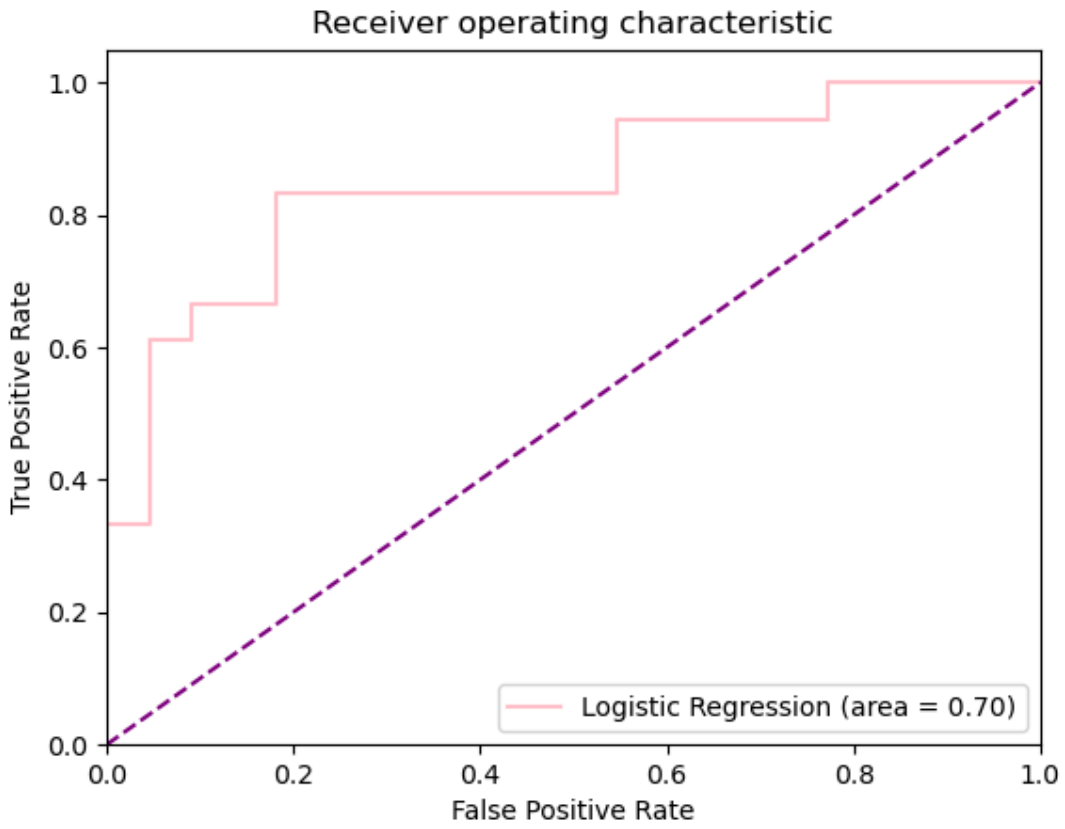
The Case

The email dataset accompanying this activity guide, 'ActivityGuide-Week13_data.xlsx', contains 9 variables, namely: spam (coded as '1' if email instance is classified as spam, or '0' if otherwise - not a spam), to_multiple, image, attach, password, line_breaks, format, re_subj, and urgent_subj, with total instances (or, rows excluding the row header) of 200 email samples classified equally as a 'spam' or not. A data analyst intends to develop a spam filter using this dataset. She poses this question to himself: **Can a logistic regression model effectively classifies email messages as spam or not based on its characteristics coded as predictor variables (i.e., from 'to_multiple' to 'urgent_subj'?**

The Logistic Regression Output

Fitted Model Summary Output

Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC) Plot



Regression Function

$$P(spam) = \frac{1}{1 + e^{-(b_0 \cdot tomultiple + b_1 \cdot image + b_2 \cdot attach + b_3 \cdot password + b_4 \cdot linebreaks + b_5 \cdot format + b_6 resubj + + b_7 \cdot urgentsubj + a)}}$$

where:

$P(spam)$ = Probability of being spam
a: model intercept = 0.9877532938524807
b_0: regression coefficient for `to_multiple` = -2.287615166270323
b_1: regression coefficient for `image` = -1.5864431119064175
b_2: regression coefficient for `attach` = -1.0600952562937187
b_3: regression coefficient for `password` = -0.6489497718099202
b_4: regression coefficient for `line_breaks` = -0.5684230708939143
b_5: regression coefficient for `format` = -4.130615983600187e-05
b_6: regression coefficient for `re_subj` = 0.5663764554627378
b_7: regression coefficient for `urgent_subj` = 0.6749611018286835

Accuracy Score

The final accuracy score of the sk-learn logistic regression model is equal to 0.675 or 67.50%.

Confusion Matrix

		Actual Values	
		Positive	Negative
Predicted Values	Positive	10	12
	Negative	1	17

Classification Report

	precision	recall	f1-score	support
0	0.91	0.45	0.61	22
1	0.59	0.94	0.72	18
accuracy			0.68	40
macro avg	0.75	0.70	0.66	40
weighted avg	0.76	0.68	0.66	40

Statsmodels Summary

Logit Regression Results

Dep. Variable:	spam	No. Observations:	160
Model:	Logit	Df Residuals:	151
Method:	MLE	Df Model:	8
Date:	Fri, 03 May 2024	Pseudo R-squ.:	0.3204
Time:	23:29:29	Log-Likelihood:	-75.338
converged:	False	LL-Null:	-110.85
Covariance Type:	nonrobust	LLR p-value:	3.062e-12

	coef	std err	z	P>|z|	[0.025	0.975]
const	1.3540	0.396	3.419	0.001	0.578	2.130
to_multiple	-1.6458	0.873	-1.885	0.059	-3.357	0.065
image	-1.7557	1.718	-1.022	0.307	-5.122	1.611
attach	1.0422	1.117	0.933	0.351	-1.147	3.231
password	-28.3666	3.75e+05	-7.57e-05	1.000	-7.35e+05	7.35e+05
line_breaks	7.961e-05	0.001	0.150	0.881	-0.001	0.001
format	-0.9528	0.467	-2.042	0.041	-1.867	-0.038
re_subj	-3.5365	0.783	-4.514	0.000	-5.072	-2.001
urgent_subj	23.4247	2.55e+04	0.001	0.999	-5.01e+04	5.01e+04

Narrative Text

Exploratory Data Analysis

Import the `pandas` library for data manipulation and analysis.

```python
import pandas as pd
```

Load the excel sheet into a dataframe with `panda`'s `read_excel` function.

```python
# Import dataset
path =
r"C:\Users\daeni\Desktop\LOVE\Academics\CMSC178DA\CMSC178-Data-Analysis-Proj
ects\Activity\Logistic Regression\data.xlsx"
df = pd.read_excel(path)
df.head()
```

	spam	to_multiple	image	attach	password	line_breaks	format	re_subj	urgent_subj
0	1	0	0	0	0	364	1	0	0
1	1	0	0	0	0	71	1	0	0
2	1	0	0	0	0	35	1	0	0
3	1	0	0	2	0	22	0	0	0
4	1	0	0	0	0	2	0	0	0

Modify the columns in the data frame under image, attach, and password to have dichotomous values wherein if the value is greater than 1, it will be just set to 1.

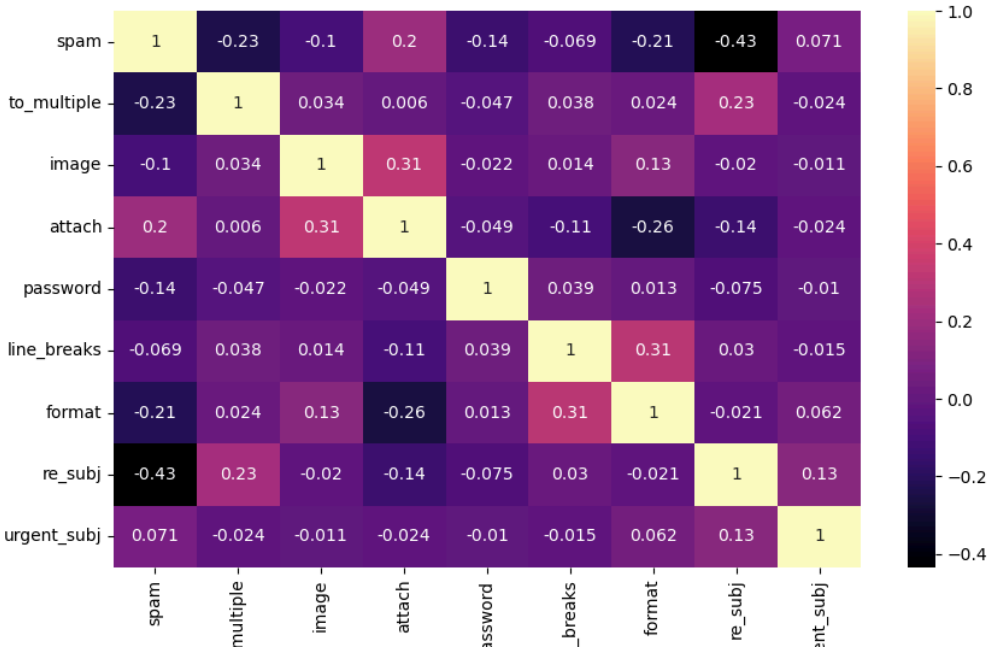
```python
df[['image', 'attach', 'password']] = df[['image', 'attach',
'password']].applymap(lambda x: 1 if x > 1 else x)
```

Import `matplotlib` and `seaborn` for data visualization techniques and methods.

```python
import matplotlib.pyplot as plt
import seaborn as sns
```

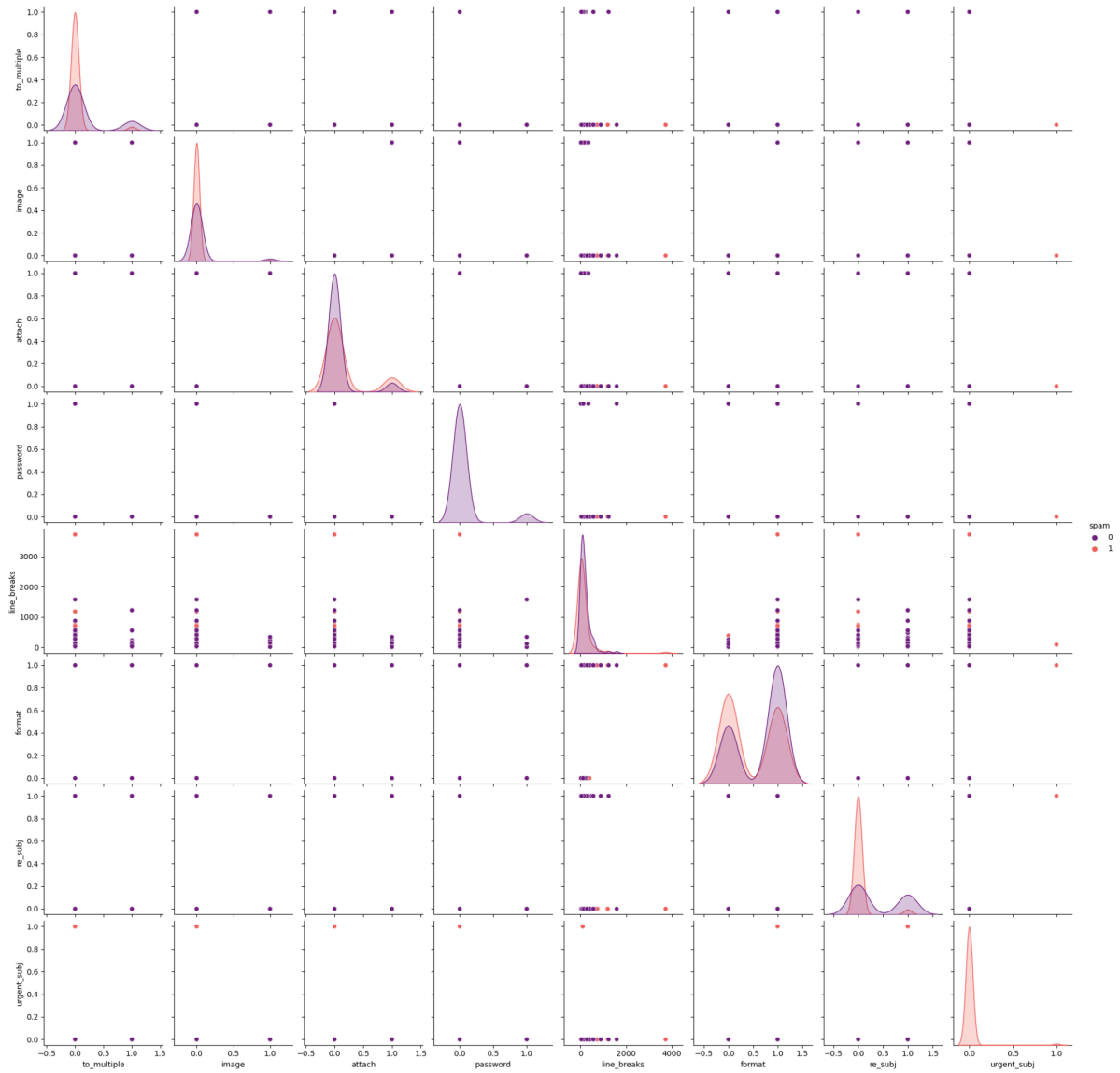
To illustrate the pairwise correlation between variables, `seaborn`'s `heatmap()` function was used. This requires correlation data, which can be obtained using the `corr()` function. By passing this correlation data to the heatmap and setting annotation to `true`, the numerical values representing the correlation between variables can be displayed.

```python
# So that column names are shown
fig, ax = plt.subplots(figsize=(10, 6))

sns.heatmap(df.corr(), annot=True, cmap='magma')

plt.show()
```



Additionally, a pairplot is also utilized to show the pairwise relationship between these variables.

```python
sns.pairplot(df, hue='spam', palette='magma')
plt.show()
```



SciKit Learn Logistic Regression Model Development

Import the essential tools from the scikit-learn library to construct the `LogisticRegression` model. This model will be trained to predict outputs. To partition the data frame `df` into training and testing sets, the `train_test_split` tool was imported.

```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

Split the dataset into training and testing variables for both the `features` and the `label`. The test size is set to 20% of `df`, indicating that 80% of the data will be used for training the model. The random seed was set to `0` for reproducibility.

```python
# Target value
label = df.pop('spam')

# Features to predict target variable
features = df

# Split the dataset
x_train, x_test,  y_train, y_test = train_test_split(features, label,
test_size=0.2, random_state=0)
```

Instantiate the `LogisticRegression()` model into a variable named `model`.

```python
model = LogisticRegression()
```

The `fit` function trains a model by establishing the relationship between the `features` and the `label` variable. It utilizes features such as `to_multiple, image, attach, password, line_breaks, format, re_subj, and urgent_subj` to understand their impact on determining whether an email is spam or not. This enables the model to generate predictions for new data by leveraging the patterns it identifies within the training dataset.

```python
model.fit(x_train, y_train)
```

The trained logistic regression model offers the model intercept `a` from its `intercept_` attribute and coefficients (`b0, b1, b2, b3, b4, b5, b6, b7`) from `coef_`, corresponding to the regression coefficients from features,`to_multiple, image, attach, password, line_breaks, format, re_subj, and urgent_subj`, respectively.

```python
# Prints model intercept
print(f"Values\na: {model.intercept_[0]}")
# Prints regression coefficient of feature variables
for i in range(8):
    print(f"b{i}: {model.coef_[0][i]}")
```

```
Values
a: 0.9877532938524807
b0: -1.0600952562937187
b1: -0.5684230708939143
b2: 0.5663764554627378
b3: -1.5864431119064175
b4: -4.130615983600187e-05
b5: -0.6489497718099202
b6: -2.287615166270323
b7: 0.6749611018286835
```

SciKit-Learn Logistic Regression Model Evaluation

Scikit-learn's metric tools were imported to further evaluate the previously created `model`. The `accuracy_score` function calculates the accuracy of correct predictions, which is particularly suitable for classification models like the one constructed. In contrast, the `confusion_matrix` assesses the model's performance by indicating how many samples were correctly or incorrectly classified by the algorithm in each class. Finally, the `classification_report` was employed to gauge the quality of predictions generated by the model.

```python
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report
```

The `accuracy_score()` function was used to determine the accuracy of the `model` based on the predicted values obtained from `x_test` which is `y_hat` relative to `y_test`. It evaluates the proportion of correctly classified instances relative to the total instances in the test set.

```python
# Make predictions with the model using the test data frame
y_hat = model.predict(x_test)

accuracy = model.accuracy_score(y_test, y_hat)

print(f"Accuracy of the scikit-learn logistic regression model given the
test values = {accuracy}")
```

```
Accuracy of the logistic regression model given the test values = 0.675
```

The `confusion_matrix()` function provides a summary of the model's predictions on the test dataset `y_hat` in comparison to the actual labels `y_test`. This is presented in the grid map as follows:

```python
cm = confusion_matrix(y_test, y_hat)
print ("Confusion Matrix\n", cm)
```

```
Confusion Matrix
 [[10 12]
 [ 1 17]]
```

Utilizing similar inputs as the previous functions, the `classification_report()` function accepts `y_test` and `y_hat` as parameters. It then presents key metrics of the `model`'s performance, providing insights into how well it performed in distinguishing between spam (`1`) and non-spam (`0`) content.

```python
cr = classification_report(y_test, y_hat)
print("Classification Report\n", cr)
```

	precision	recall	f1-score	support
0	0.91	0.45	0.61	22
1	0.59	0.94	0.72	18
accuracy			0.68	40
macro avg	0.75	0.70	0.66	40
weighted avg	0.76	0.68	0.66	40

StatsModels Logistic Regression Model Development

To support the findings from the scikit-learn model, a new model was developed using the statsmodels API and tools imported as follows for statistical modeling and evaluation. It serves an additional function of getting p-values for the individual features and other values that would be hard to derive from the `scikit-learn` library solely.

```python
import statsmodels.api as sm
```

Prepare the data and extract the p-values for the logistic regression model name `sm_model` built with `statsmodels`. It first adds a constant term (intercept) from `x_train` using `sm.add_constant` since `statsmodels` doesn't include it by default. Then, it builds the model using `sm.Logit` with the training set.

```python
# Add constant to training data
x_train_const = tools.add_constant(x_train)
# Train the logistic regression with the training data
sm_model = sm.Logit(y_train, x_train_const).fit()
```

Statsmodels Logistic Regression Model Evaluation

The code extracts a summary of the model results using `model.summary()`.

```python
print(sm_model.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                   spam   No. Observations:                  160
Model:                          Logit   Df Residuals:                      151
Method:                           MLE   Df Model:                            8
Date:                Fri, 03 May 2024   Pseudo R-squ.:                  0.3204
Time:                        23:29:29   Log-Likelihood:                -75.338
converged:                      False   LL-Null:                       -110.85
Covariance Type:            nonrobust   LLR p-value:                 3.062e-12
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.3540      0.396      3.419      0.001       0.578       2.130
to_multiple   -1.6458      0.873     -1.885      0.059      -3.357       0.065
image         -1.7557      1.718     -1.022      0.307      -5.122       1.611
attach         1.0422      1.117      0.933      0.351      -1.147       3.231
password     -28.3666   3.75e+05  -7.57e-05      1.000   -7.35e+05    7.35e+05
line_breaks  7.961e-05      0.001      0.150      0.881      -0.001       0.001
format        -0.9528      0.467     -2.042      0.041      -1.867      -0.038
re_subj       -3.5365      0.783     -4.514      0.000      -5.072      -2.001
urgent_subj   23.4247   2.55e+04      0.001      0.999   -5.01e+04    5.01e+04
==============================================================================
```

Model Evaluation through Visualization

A graphical representation commonly used to evaluate a model is the receiver operating characteristic (ROC) curve, which illustrates the trade-off between the true positive rate (TPR) and the false positive rate (FPR) for different classification thresholds. Another essential statistic for assessing binary classification models is the area under the ROC curve (AUC). These metrics are imported here to visually assess the accuracy of the model by observing the TPR and FPR.

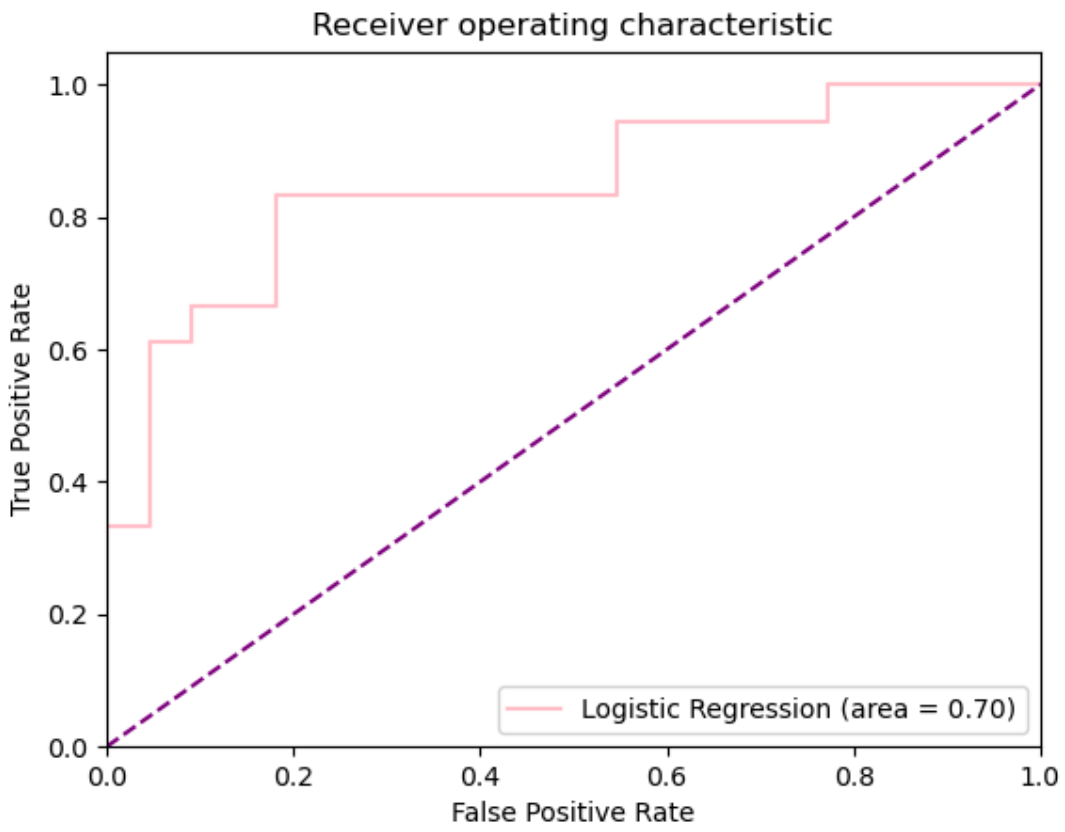
```python
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

Using the predicted values from the test set, both the actual `y_test` values and the predicted `y_hat` values are used to calculate the score of the model through the `roc_auc_score()` function passed onto `logit_roc_auc`. The next line of code takes the test labels (`y_test`) and the predicted probabilities of the spam class (`model.predict_proba(x_test)[:,1]`) as inputs. This process yields three outputs: `fpr` values, `tpr` values, and the corresponding `thresholds` used in the ROC curve calculation.

```python
logit_roc_auc = roc_auc_score(y_test, y_hat)
fpr, tpr, thresholds = roc_curve(y_test, model.predict_proba(x_test)[:,1])
```

Finally, the code line illustrates the relationship between the true positive rate and the false positive rate, while visualizing the area under the curve for logistic regression.

```python
logit_roc_auc = roc_auc_score(y_test, y_hat)
fpr, tpr, thresholds = roc_curve(y_test, model.predict_proba(x_test)[:,1])
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc, color='pink')
plt.plot([0, 1], [0, 1],'r--', color='purple')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```



Interpretation

Is the fitted logistic regression model effective in classifying email messages as spam or not? Justify.

The model is not as effective in classifying email messages as spam or not given that it only has an accuracy score of 67.5%, a relatively poor performance for a model. Additionally, the confusion matrix provides a detailed breakdown of the proportion of true positives, true negatives, false positives, and false negatives which shows the following:

```
Confusion Matrix
 [[10 12]
 [ 1 17]]
```

From the above results, it's evident that while the model correctly identified 10 instances of spam emails (true positives) and accurately classified 17 non-spam emails (true negatives), it also exhibited a notable number of false positives. These false alarms, where the model incorrectly labeled non-spam emails as spam, outnumbered the true positives. Additionally, there was only one case of a false negative, indicating a relatively low rate of missed spam emails. This means that the model has a hard time determining accurately whether the email is actually spam. This can further be explored through the classification report shown below:

	precision	recall	f1-score	support
0	0.91	0.45	0.61	22
1	0.59	0.94	0.72	18
accuracy			0.68	40
macro avg	0.75	0.70	0.66	40
weighted avg	0.76	0.68	0.66	40

The precision value indicates that of all the emails predicted to be spam by the model, only 59% were actually spam. Conversely, the high recall value of 94% for spam predictions suggests that the model correctly identified 94% of the actual spam emails. However, the F1 score of 72% for spam indicates a relatively poor model performance, as it is not that close to 100%. This means that the model is confident about the few positive instances it predicts but might miss a substantial number of true positives.

Examining the support values reveals that there were only 22 non-spam emails and 18 spam emails in the test dataset, with the latter being fewer than the former. This suggests

an imbalance in the dataset, with fewer instances of spam compared to non-spam emails. Consequently, this imbalance may explain the higher true negative rate observed earlier, as well as the higher precision for non-spam predictions (91%) compared to spam predictions (59%).

After analyzing the Receiver Operating Characteristic (ROC) curve, it's evident that the Area Under the Curve (AUC) value is 70%. The ROC curve trending towards the upper left corner is generally considered favorable, indicating better accuracy, as higher sensitivity and specificity are achieved.

However, despite the promising direction of the ROC curve, the AUC value falls short of the ideal 100%, settling at 70%. This discrepancy indicates that while the model exhibits some discriminatory power, it does not perform as well as desired in distinguishing between spam and non-spam emails.

In conclusion, the model's effectiveness in classifying emails as spam or non-spam is compromised by its suboptimal performance across various metrics. Further training of the model with a better and bigger dataset may be necessary to improve its classification accuracy.

Which of the predictor variables significantly influence the classification of email messages as spam or not? Identify. Justify.

The analysis of the dataset reveals that the most influential predictor of whether an email is classified as spam or not is if it is an email sent to multiple people (`to_multiple`). This conclusion is drawn from its regression coefficient, which has the largest absolute value $|-2.287615166270323|$ among all the predictor variables.

The negative coefficient associated with the variable `to_multiple` suggests an inverse relationship which means that when an email is flagged as sent to many, the likelihood of those emails being classified as spam tends to decrease.

The following shows the ranking the variables from least to greatest in terms of their influence on spam classification:

1. b_0: $tomultiple = |-2.287615166270323|$
2. b_1: $image = |-1.5864431119064175|$
3. b_2: $attach = |-1.0600952562937187|$
4. b_7: $urgentsubj = |0.6749611018286835|$
5. b_3: $password = |-0.6489497718099202|$
6. b_4: $linebreaks = |-0.5684230708939143|$

7. b_6: $resubj = |0.5663764554627378|$
8. b_5: $format = | - 4.130615983600187e - 05|$