


```
[46]: from sklearn.linear_model import LogisticRegression
      from sklearn.model_selection import train_test_split
      from sklearn.preprocessing import StandardScaler
```

Split the dataset into independent variables (x) and the dependent (target) variable (y)

```
In [48]: # split into target and independent variables
x = df_cleaned[['word_count', 'sentence_count', 'avg_sentence_len', 'polarity', 'negativity', 'neutrality', 'subjectivity']]
y = df_cleaned['is_human'] # target
```

Additionally, we will also scale the x values using sci-kit's StandardScaler. This scales each x value so that they have the same scale and dimension, which prevents differently-scaled variables from introducing biases into the modelling.

```
In [49]: # init scaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(x)
```

Split data into testing (20%) and training (80%) sets using sklearn's train_test_split

```
In [50]: x_train, x_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.2, random_state = 42)
```

Initialise a logistic regression model, and train it on the training set

```
In [71]: LR = LogisticRegression(max_iter=10000)
LR.fit(x_train, y_train)
```

```
Out[71]: LogisticRegression
LogisticRegression(max_iter=10000)
```

Use the logistic regression model to predict the target (y) values.

Also, we will generate the model's predicted probability of each class prediction

```
In [92]: # generate prediction
y_prediction = LR.predict(x_test)

#generate probabilities
y_prob = LR.predict_proba(x_test)
# keep only the probability of the positive class
y_prob = y_prob[:, 1]
```

4. Evaluation

First, install the matplotlib and numpy packages, and import modules

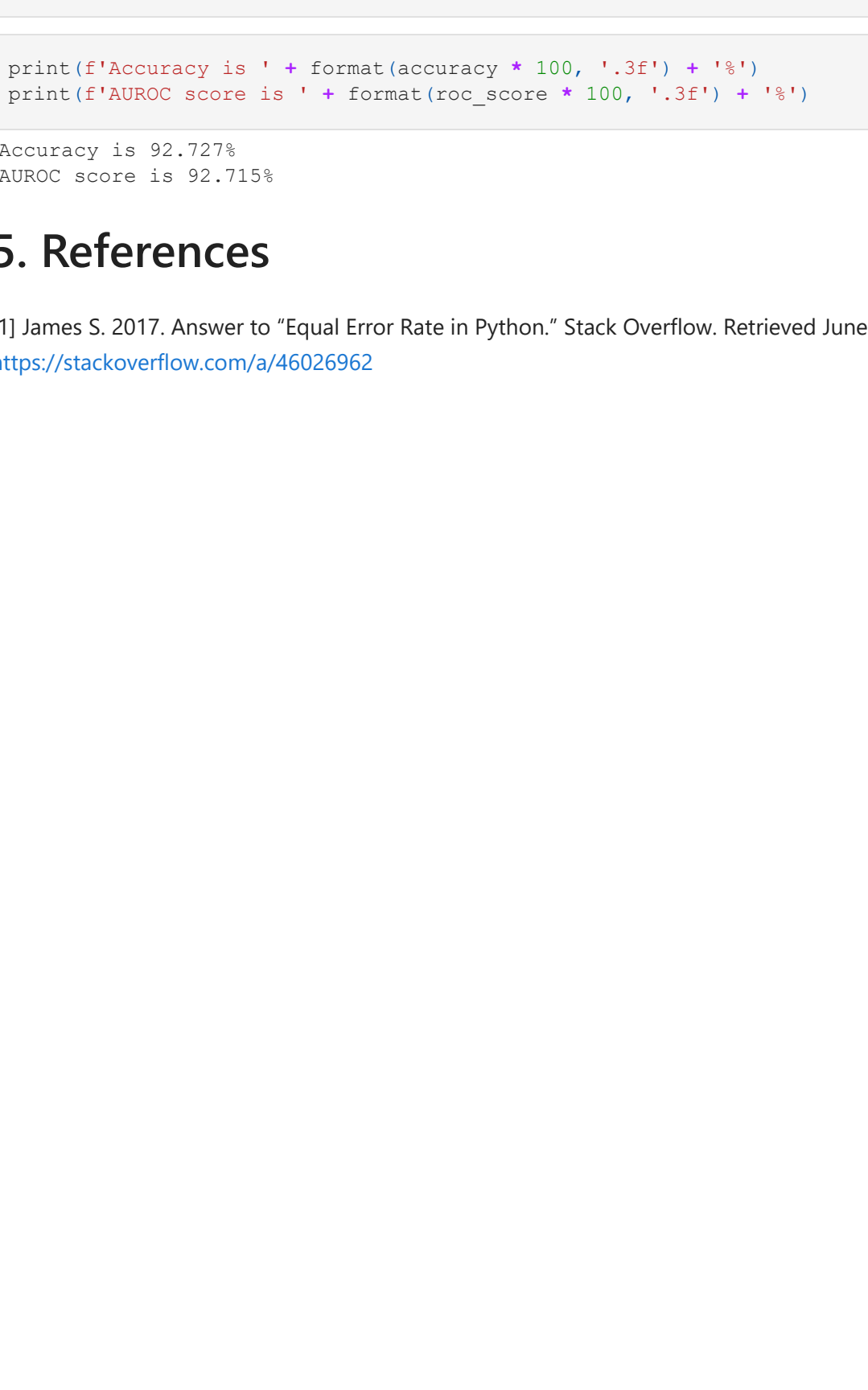
```
In [ ]: # !pip install matplotlib
        # !pip install numpy
```

```
In [85]: from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
import numpy as np
```

4.1 False Positive Rate, True Positive Rate, and Equal Error Rate

First, we will plot the model's confusion matrix

```
In [89]: #create confusion matrix
con_matrix = confusion_matrix(y_test, y_prediction)
# Create confusion matrix display
cm_display = ConfusionMatrixDisplay(con_matrix, display_labels=['gpt', 'human'])
#plot and show confusion matrix
cm_display.plot()
plt.show()
```



Next, we will use the values in the confusion matrix to calculate the model's False Positive Rate (FPR) and True Positive Rate (TPR)

```
In [90]: # compute FPR and TPR (recall)
false_pos_rate = con_matrix[0][1] / (con_matrix[0][0] + con_matrix[0][1])
true_pos_rate = con_matrix[1][1] / (con_matrix[1][0] + con_matrix[1][1])

print(f'FPR is ' + format(false_pos_rate * 100, '.3f') + '%')
print(f'FPR is ' + format(true_pos_rate * 100, '.3f') + '%')
```

FPR is 9.350%
TPR is 94.779%

Lastly, we will compute the model's Equal Error Rate, using the method proposed in James S. [2017]

```
In [94]: # generate roc curve, and fpr, tpr lists and the threshold
fpr, tpr, threshold = roc_curve(y_test, y_prob, pos_label = 1)
# compute false negative rate
fnr = 1 - tpr
# find the threshold where the EER occurs
eer_threshold = threshold[np.nanargmin(np.absolute((fnr - fpr)))]
#find EER
eer = fpr[np.nanargmin(np.absolute((fnr - fpr)))]

print(f'EER is ' + format(eer * 100, '.3f') + '%')
```

EER is 7.317%

4.2 Accuracy and ROC Score

Use scikit-learn's built-in functions to find the model's accuracy and AUROC score

```
In [87]: accuracy = accuracy_score(y_test, y_prediction)
roc_score = roc_auc_score(y_test, y_prediction)
```

```
In [88]: print(f'Accuracy is ' + format(accuracy * 100, '.3f') + '%')
print(f'AUROC score is ' + format(roc_score * 100, '.3f') + '%')
```

Accuracy is 92.727%
AUROC score is 92.715%

5. References

[1] James S. 2017. Answer to "Equal Error Rate in Python." Stack Overflow. Retrieved June 7, 2023 from <https://stackoverflow.com/a/46026962>