13.Logistic Regression

We will not start to test and train our machine learning model using our newly organized data. First we need to make sure we have imported some important libraries to run some modules we will use. Make at the top of your code to import sklearn model_selection , sklearn.linear model LogisticRegression, sklearn preprocessing, pickle, sklearn.metric plot_confusion_matrix Your code should look like this:

```
import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import model_selection, preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import ConfusionMatrixDisplay
```

Next we will be changing all of the organization labels to bots as logistic regression can only take in values of 1 or 0. therefore for all the labels that we made to equal 2 we will change to 0. Then we shall create a new dataframe using our organized csv already created to use for our regression. For logistic regressions we only need the id and lavels so we use the pd method drop to drop all other columns besides these two. That is assigned to x1. We have to create 4 variables x_train, x_test, y_train, y_test and use the model selection train_test_split method to train and test our data. We then utilize the preprocessing.scale method to hone in our models then we call the Logistic Regression method that we created and use model.fit. Then we create a result variable that holds the model.score() method value received from the x_test_scaled variable and y_test variable. We then print out the accuracy of our model. Your code should look like this:

```
def logistic_regression(input_df, print_result=False):
    x = input_df.drop(["spammer", "label"], axis=1).values
    if "id" in input_df.columns:
        x = input_df.drop(["id", "CAP", "spammer", "label"], axis=1).values
    y = input_df["label"].values
    x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=0.30, random_state=100)

    x_train_scaled = preprocessing.scale(x_train)
    x_test_scaled = preprocessing.scale(x_test)

    model = LogisticRegression()
    model.fit(x_train_scaled, y_train)
    result = model.score(x_test_scaled, y_test)

    pickle.dump(model, open(f'log_reg_eng.dat', 'wb'))

    if print_result:
        print("Accuracy: %.2f%%" % (result * 100.0))
    return result * 100
```

14.Removing Outliers

Your code should be running however we want to increase the accuracy as an accuracy of 83% is no good. There are some things we are going to do in order to increase the accuracy. One of them Is removing outliers from your data. Outliers are datapoints that are way outside the mean (average). The easiest way to do this is by utilizing the z score which corresponds to the standard deviation. Standard deviation is a term used in statistics to refer to the measurement of distance data points have in relation to the mean (average). In order words it’s the variance/dispersion of a set of values. It is given by this equation.

Formula

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

σ = population standard deviation
N = the size of the population
x_i = each value from the population
μ = the population mean

The z-score is measured in standard deviation which shows the relationship between an individual data point and the mean (average) of the other data points. Z-scores are used to determine if an individual data point is too far outside of the mean (average) and are known as outliers. Z-scores are found using this formula.

Formula

$$Z = \frac{x - \mu}{\sigma}$$

Z = standard score
x = observed value
μ = mean of the sample
σ = standard deviation of the sample

Note: For this project you do not need in depth knowledge of Statistics however understanding the basics help when analyzing your final results. If you are having trouble understanding Z-scores and Standard deviation I would recommend watching this video on the topic.
https://www.youtube.com/watch?v=2tuBREK_mgE

We want to create a function named remove_outliers() that takes in a z-score and filters the data set based on whether a twitter account has a z-score above the targeted z-score. The z-score we will be using is 2.5. We want to grab the training set and separate them by category while also turning the organizations to bots. We then filter the dataframe of values containing a z-score higher than 2.5 for human and non-human then we concatenate the two dataframes. To utilize the z-score you will need to import scipy as sklearn does not currently have the stats.score() function. It should look this:

```python
def remove_outliers(zscore, input_df, output_path=None):
    human_df = input_df[input_df['label'] == 0]
    bot_df = input_df[input_df['label'] == 1]

    human_df = human_df[np.abs(find_zscore(human_df.loc[:, "astroturf":"self-declared"]) < zscore).all(axis=1)]
    bot_df = bot_df[np.abs(find_zscore(bot_df.loc[:, "astroturf":"self-declared"]) < zscore).all(axis=1)]

    output_df = pd.concat([human_df, bot_df])
    output_df = output_df.loc[:, "astroturf":]

    if output_path is None:
        return output_df
    else:
        output_df.to_csv(output_path)
        return output_df
```

15.Graphing Accuracy vs Z-score

We also want to graph the z-score vs the accuracy to see which z-score gives us the highest accuracy. To do this we will set our standard deviation to the highest feasible amount we cant and using a while loop decrement it and graph the accuracy for each z-score. We obtain the accuracy from the Logistic Regression function we have already made however beforehand we pass in our standard deviation (z-score) into the remove outliers function before the Logistic Regressions runs. It should look like this :

```python
def best_zscore(zscore, lang, input_path, show=False):
    input_df = orgs_to_bots(input_path)
    zscores = []
    accuracies = []

    while zscore >= 0:
        zscores.append(zscore)
        df = remove_outliers(zscore, input_df)
        result = logistic_regression(df)
        accuracies.append(float("{:.2f}".format(result)))
        zscore -= 0.1

    plt.plot(zscores, accuracies)
    plt.xlabel('zscore')
    plt.ylabel('accuracy (%)')
    plt.suptitle(f"Zscore and Accuracy ({lang})")

    plt.savefig(fr'C:\Users\Rachel\Documents\Twitter Data\zscore_{lang}_test')
    if show:
        plt.show()

```

- The code resembles the one used in the video and I liked how in the document, he also went into details and also utilizies the formula to further explain the outliers.
-