**Introduction:**

This project is to use neural network to analyze and determine whether a tweeter is male or female. Here we will use sci-kit along with the nltk to train and test the neural network.

**Precprocessing:**

Feature Selection and Engineering:

The following were selected after removing null columns and removing columns determined unnecessary in EDA:

1. gender
2. fav\_number
3. Text -> PredictedbyNLP
4. tweet\_count

Feature engineering was done by predicting the the gender using the comments and tweets of the user. This predicted gender was used as a dependent data for the neural network.

Label Encoding:

Only the gender field was required to be encoded, which was done via the LabelEncoder() builtin function of the sci-kitlearn package of python.

**EDA:**

1. First we see the counts of \_unit\_state, \_golden, \_trusted\_judgments all were constant other than the gold members which are negligible in numbers.
2. Next we see the relation of fav\_number against the gender. The swarm plot does not show a very large difference, but when calculating the average we can see that women have a tendency to pick larger numbers compared to men.
3. We see that almost all have zero retweets, while some have one retweet, others are low enough in numbers to be considered exception, but the swarm plot shows that females have higher retweet counts
4. When we come to the tweet counts, we see that males have a higher average in number of tweets when compared to the females.

**Project Summary:**

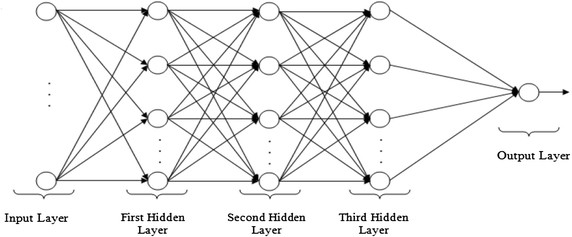
First step after importing the data was to remove all the unnecessary fields and use LabelEncoder() on the the gender field, after removing the gold members. Then we split the dataset to have equal number of male and female records to prevent biasing of the model.

Then we transverse through each comment in the database, and clean each comment by finding the root word (using PortStemmer()) and checking if they are part of the stop words stored in the inbuilt library of nltk package of python. These cleaned comment are added to a new field called ‘CleanedComments’. These comments are then used to perform gender prediction and the prediction of these are used in the training of the neural network. This prediction is done by using the inbuilt naive bayers in the sci-kitlearn python package. Here we convert the ‘CleanedComments’ into a sparse matrix using the CountVectorizer() from sklearn. Then we train the Naive Bayes model using the sparse matrix as the only dependent feature and then having it predict the outcomes of the same sparse matrix. Though this may seem unecessary, we need to do this as the neural networl will not be able to efficiently use the comments of the tweeters to predict their genders. This is done in order to simulate the prediction of the neural network through NLP

After the prediction based off of the comments are added to the dependent features, we split the database into train and test sets. We then form a StandarScaler() model to scale the data for uniform evaluation. We train the scaler with the input training data and transform the train and test data for a better accuracy.

Now we create a neural network model to predict the gender, this time with the prediction of the Naive Bayer’s model. We are using the Multilayer Percepton classifier model, with a hyperbolic tan as the activation function and 3 hidden layers with 8 perceptons on each layer. We used an stochastic gradient-based optimizer provided within the inbuilt model of MLPClassifier() from sklearn since it had a higher accuracy and was relatively faster than the other optimizer when a few trials were conducted. The model was trained using the training data and the prediction of the test data was compared to the actual results, and the accuracy was printed.

The structure of the network comes out similar to:



than with 8 perceptons in each hidden layer with 4 in the input layer and 2 in the output layer.

The output layer will give probabilities of each result and these probabilities will determine the result.

We come to see that the accuracy is always above 88%, which is better than expected for a relatively small dataset like the one used.