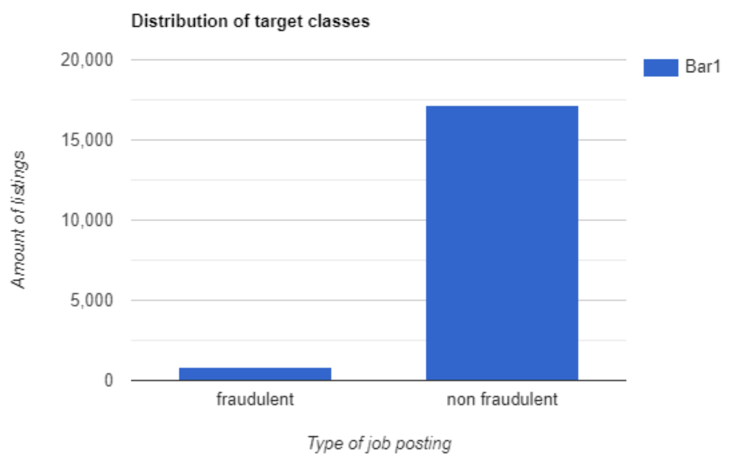
Text classification

For this homework, I found a dataset on Kaggle that contained data of fake and real job postings. The dataset contained 800 fraudulent postings, and 17,200 real postings. The following graph shows this distribution.



This data set contains plenty of information including type of job, salary, and many other details. Because I think the description of the job should contain many nuances that allow a model to pick up on its legitimacy, I used descriptions as the data I would train my model on.

Ideally the model should be able to predict whether or not a posting is fraudulent based off of the description.

Here I am just cleaning the code, and then setting it up to be used by machine learning algorithms. The test data will be 20% of the data, and the train data will be the rest of it.

```
import pandas as pd
df = pd.read_csv('../content/joblistings.csv', header=0, usecols=[6,17], encoding='latin-1')
```

```
from nltk.corpus import stopwords
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=list(stopwords))
X = df.description
y = df.fraudulent
X, y = df.description.fillna(' '), df.fraudulent

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, random_state=1234)

X_train.shape
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

Here I am testing out the Naive Bayes algorithm on my data.

```
from sklearn.naive_bayes import MultinomialNB
import math
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, y_train)


# make predictions on the test data
pred = naive_bayes.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred))
print('recall score: ', recall_score(y_test, pred))
print('f1 score: ', f1_score(y_test, pred))
```

```
    accuracy score:  0.947986577181208
    precision score:  0.0
    recall score:  0.0
    f1 score:  0.0
```

At first glance, an accuracy score of 94.79 is pretty good, but a precision score of 0.0 shows that none of the fraudulent job postings were correctly classified as fraudulent. Since the Multinomial class considers counts of words, it might not be the best predictor of whether or not a listing is fraudulent.

This time we will use the Binomial classifier which only looks at whether a word is present or not to see if that will help.

```
from sklearn.naive_bayes import BernoulliNB

naive_bayes2 = BernoulliNB()
naive_bayes2.fit(X_train, y_train)
pred = naive_bayes2.predict(X_test)

print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred))
print('recall score: ', recall_score(y_test, pred))
print('f1 score: ', f1_score(y_test, pred))
```

```
    accuracy score:  0.9569351230425056
    precision score:  0.8297872340425532
    recall score:  0.21081081081081082
    f1 score:  0.33620689655172414
```

These results look much better, the accuracy and precision are all improved from the last test.

Next, I will look at how well the Logistic Regression algorithm does on my dataset.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, log_loss


classifier = LogisticRegression(solver='lbfgs', class_weight='balanced')
classifier.fit(X_train, y_train)

pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred))
print('recall score: ', recall_score(y_test, pred))
print('f1 score: ', f1_score(y_test, pred))
probs = classifier.predict_proba(X_test)
print('log loss: ', log_loss(y_test, probs))
```

```
    accuracy score:  0.9628076062639821
    precision score:  0.6007751937984496
    recall score:  0.8378378378378378
    f1 score:  0.6997742663656884
    log loss:  0.17994407102638446
```

The Logistic Regression model was much better in terms of accuracy, but worse precision. This means that the Logistic Regression model was better at identifying when a listing was not fraudulent, but worse at identifying when a listing was actually fraudulent.

Next, lets look at how the neural network fares. I tried out many nodes and layers, and the best results came from one layer with 10 nodes.

```
from sklearn.neural_network import MLPClassifier
classifier2 = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(10), random_state=1)
classifier2.fit(X_train, y_train)
```

```
    ▾                        MLPClassifier
    MLPClassifier(alpha=1e-05, hidden_layer_sizes=10, random_state=1,
                  solver='lbfgs')
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
pred = classifier2.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred))
print('recall score: ', recall_score(y_test, pred))
print('f1 score: ', f1_score(y_test, pred))
```

```
     accuracy score:  0.977069351230425
     precision score:  0.8159509202453987
     recall score:  0.7189189189189189
     f1 score:  0.7643678160919539
```

Looking at the statistics, it seems that the neural network algorithm worked the best overall, with the best accuracy and a good precision score. Although the binomial NB algorithm had the highest precision, it also had a very low recall score, suggesting that the algorithm also gave many false positives. Because the f1 score of the neural network algorithm was much higher than the other algorithms, and because both recall and precision matter in this case, I would rank the algorithms using their f1 scores. This means that the most effective would be the neural networks, followed by the logistic regression, followed by the binomial NB, and the worst one would be the multinomial NB.

✓ 0s completed at 8:17 PM