Report (z5309451)

Preparing Data

To deal with the categorical data i used the One-Hot-Encoder which was able to split the categories into their own numerical value. Compared to other methods of encoding, the One-Hot Encoding is by far better as it does not place differing weightings on each categorical value and treats them equally.The concatenation of the encoded data as well as the encoding could be done with the get_dummies() function. I used keywords to highlight and find the columns with categorical data.

In [ ]:
```
for col in col_list:
        if ('NAME' in col) or ('FLAG' in col) or ('TYPE' in col) or ('CODE_GENDER' i

            adj_cols.append(col)
    df = pd.get_dummies(df, columns=adj_cols)
    df_test = pd.get_dummies(df_test, columns=adj_cols)
```

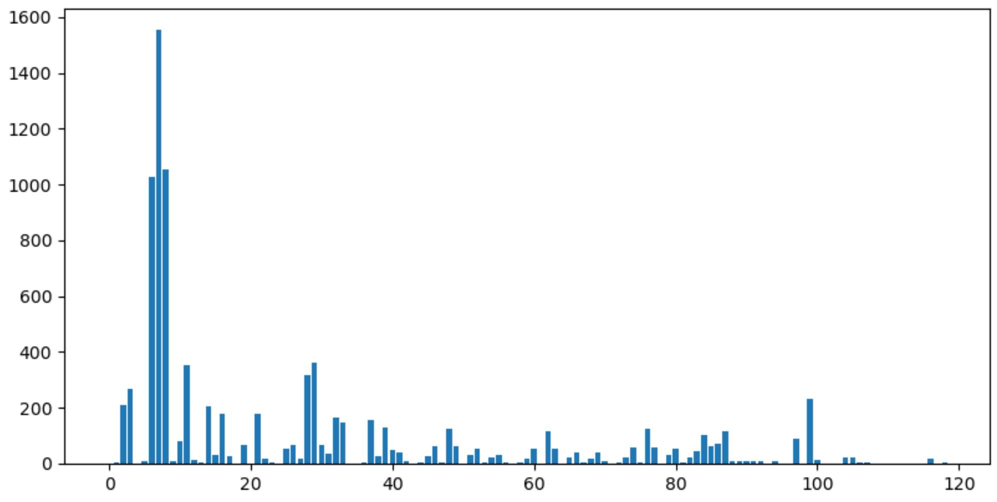
To further prepare the data I used the SimpleInputer() function which was able to fill in NAN values with an estimate based on the columns median. The median was chosen, as for columns which have been added with get_dummies() method, using median would make it so that the only a value of '1' or a value of '0' would be used. Mean would have resulted in this NAN values for categorical columns to have a differing value and hence a differing weighting.

In [ ]:
```
for column in df.columns:

        imputer = SimpleImputer(missing_values=np.nan,
                                strategy="median")
        df[column] = imputer.fit_transform(df[[column]])
```

Feature Selection

To select the correct features I employed the use of the SelectKBest Algorithm. The algorithm is able to transform the dataset to contain the n best features, based on p-value and score. A graph of the feature scores of every feature when all features where included in the dataset can be seen below.

In [ ]:
```
fs = SelectKBest(score_func=f_regression, k='all')

    fs.fit(income_x, income_y)
    income_x = fs.transform(income_x)

    income_x_t = fs.transform(income_x_t)

    for i in range(len(fs.scores_)):
        if (fs.pvalues_[i] > 0.05):
            remove_list.append(df.columns[i])

        print('Feature %s: %f' % (df.columns[i], fs.pvalues_[i]))

    # plot the scores
    pyplot.bar([i for i in range(len(fs.pvalues_))], fs.pvalues_)
    pyplot.show()
```

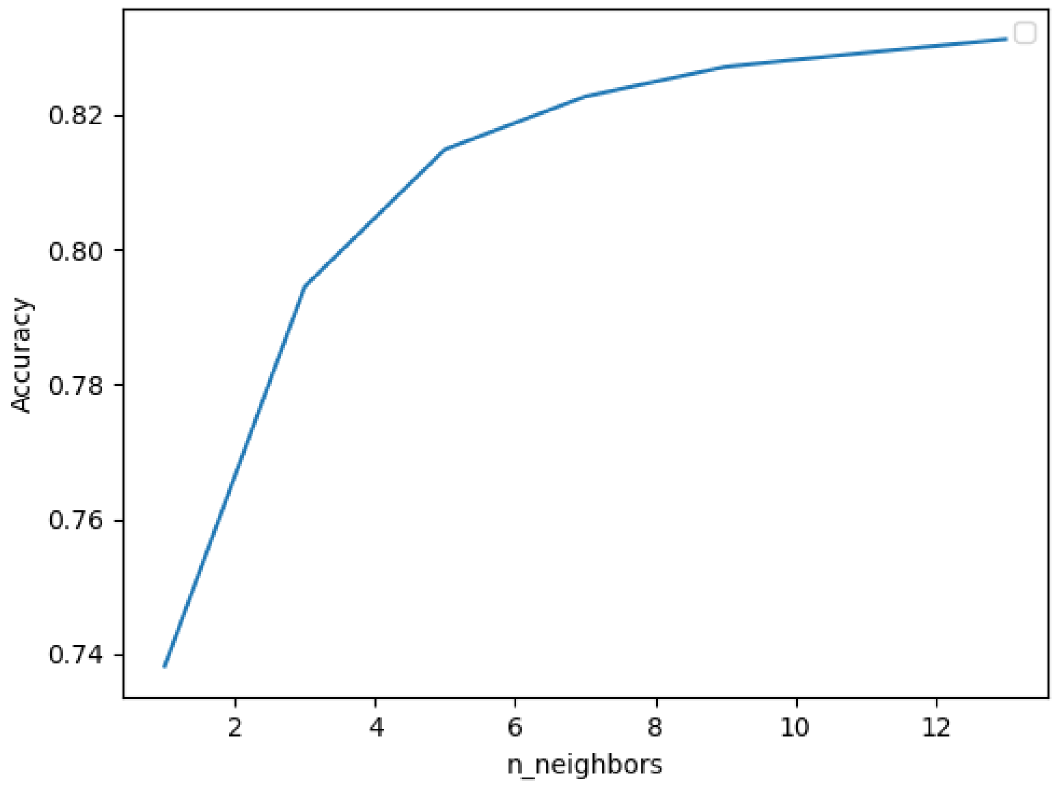


Compared to manually looking at every feature to decide if it correlated, i used a while loop to find the best value of n (in "n" best features) which gave the highest Pearson's Correlation. "n" was found to be 158 and the corresponding Pearson's Correlation was 0.54 The original Correlation on average when all features where included was 0.537638227544181. This process was mirrored when deciding for classification.

In [ ]:

```
corr = 0 ## Pearson's Correlation, starting of at zero
   k_val = 1
   for k in range(1, 253): ##Range of all features in the dataset
       income_x_train, income_y_train, income_x_test, income_y_test = create_sets(
           df, 0, df_test, k) ###This function uses SelectKBest to only select the
       model = linear_model.LinearRegression()
       model.fit(income_x_train, income_y_train)
       y_pred = model.predict(income_x_test)

       temp, _ = pearsonr(income_y_test, y_pred)
       if (temp > corr):
           k_val = k
           corr = temp
```

In order to improve the accuracy when using Classification, I tried using differing odd values of n_neighbours, as shown below.



In [ ]:

```
 for k in range(1, 14, 2):

        k_model = KNeighborsClassifier(n_neighbors=k)
        k_model.fit(income_x_train, income_y_train)
        y_pred_k = k_model.predict(income_x_test)
        temp = accuracy_score(income_y_test, y_pred_k)
        val.append(temp)
        if (temp > acc):
            acc = temp
            k_val1 = k
## best accuracy is: 0.8311666666666667
## best k is 13
```

It was found that the value of 13 produced the highest average accuracy over multiple runs. A limit of 14 was used as it is under the sqrt(NO. of Features). Values were iterated by 2 so that there was no confusion when classifying. It was also found that values past 13 produced irregular behaviour and the their accuracies slowly on average started decreasing.