MIE1624 Assignment 2 Report

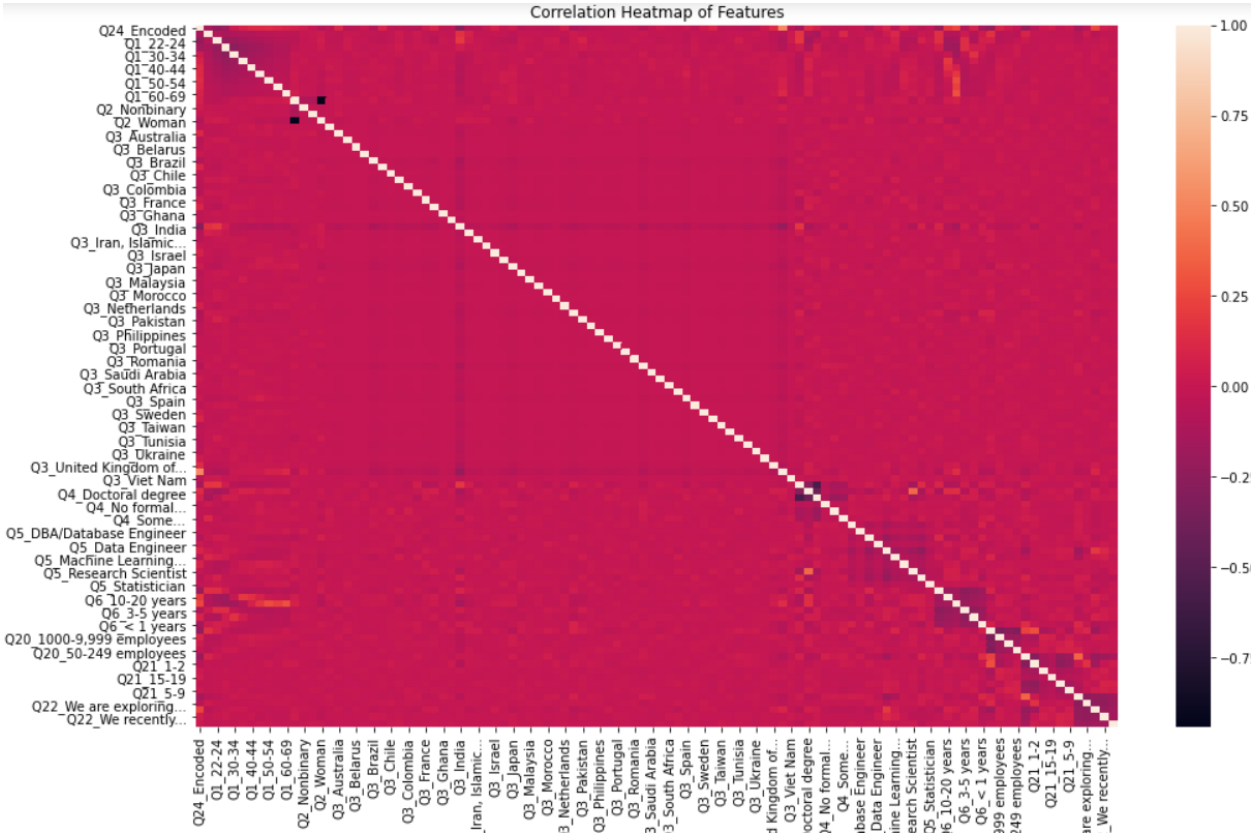
Data Cleaning

The data was first split into a training set and a test set with a ratio of 70% training data and 30% test data. For handling of missing values, I decided to drop all features that contained null entries. The reason I chose to drop features with null values is because most of these features have the majority of their entries set as null. If the null values in these features were all replaced with the same value (such as mode), then it might cause bias in model prediction. I believe that most the missing values in the data are a result of respondents either skipping a question or leaving a question blank. Thus, I believe that dropping these features with missing data will not have any drastic effect on the overall analysis of the data.

I decided to use the one-hot encoder method to convert all of the categorical features into numerical data. The reason I decided to use one-hot encoder is because some of these categorical features (such as gender and country of origin) do not have an innate order or rank. Using a one-hot encoder will preserve the unordered nature of these features.

Exploratory Data Analysis

For exploratory data analysis, I decided to visualize the order of feature importance by creating a correlation heatmap of all the features from the data cleaning step. This heatmap can be seen below:



From the covariance matrix of the features, I was able to determine that out of all the original attributes in the data, the country of origin (Q3) feature is the most related feature to yearly compensation. After some more in-depth analysis was done, I discovered that whether or not a person was from the United States had a very big impact on the person's salary bucket.

Feature Selection

Since one-hot encoder was used in the data cleaning step, we ended up with a total of 105 features. Reducing the size of the feature set by selecting only the most important features will most likely improve the performance ordinal logistic regression model, since having less features means that the model will be less likely to overfit data.

For the feature engineering step, I decided to use backward elimination to select only the important features. I used a P-value threshold of 0.05 for the regression model in my backward elimination algorithm. A portion of the features that were selected are show below:

```
The following features were selected after performing backward elimination:
Index(['Q1_18-21', 'Q1_22-24', 'Q1_25-29', 'Q1_30-34', 'Q1_35-39', 'Q1_40-44',
       'Q1_45-49', 'Q1_50-54', 'Q1_55-59', 'Q1_60-69', 'Q2_Man',
       'Q2_Nonbinary', 'Q3_Argentina', 'Q3_Australia', 'Q3_Bangladesh',


Total number of features selected: 79
```

From the above output, we are able to see that as a result of using backward elimination we were able to reduce the feature set from 105 features down to just 79 features.

Model Implementation

The ordinal logistic regression function I implemented, *ordinalLogisticRegression()*, can be found in the Python notebook. I will briefly explain how the function works:

- The function takes as input a features dataset and a target array to be used for fitting, the features dataset that we wish to predict targets for, and the hyperparameters C and penalty type.
- The function outputs a matrix that contains the probability that a datapoint belongs to a specific salary bucket for all salary buckets, and for all input datapoints. The function also outputs the predictions array which contains the salary bucket predictions for all input datapoints.
- The function works by using the inputs to perform 15 individual binary classifications, with each one corresponding to a specific salary bucket. The algorithm will predict salary bucket of a datapoint to be the one with the highest probability.

Scaling and normalization for my data is not required, since each feature will only have values of either 0 or 1 since I used the one-hot encoder in the data cleaning step.

Model Tuning

For the model tuning step, I decided to tune the hyperparameters C (inverse of regularization strength) and penalty (type of penalization). I performed a grid search with 10-fold cross-validation. The different C values I tested were 0.01, 0.1, and 1, and the two penalty types I tested were L1 and L2 penalization. A portion of the output for the accuracy and F1-score of the model across different folds can be seen below:

```
C = 0.01 , penalty = l1
Fold 1 : accuracy: 76.0705289672544 % , F1-score: 41.1451398135819 %
Fold 2 : accuracy: 80.20304568527918 % , F1-score: 42.876165113182424 %
Fold 3 : accuracy: 82.1608040201005 % , F1-score: 44.74034620505992 %
Fold 4 : accuracy: 80.10471204188482 % , F1-score: 41.81091877496671 %
Fold 5 : accuracy: 80.67885117493474 % , F1-score: 41.54460719041278 %
Fold 6 : accuracy: 80.77922077922078 % , F1-score: 42.876165113182424 %
Fold 7 : accuracy: 81.31868131868131 % , F1-score: 40.47936085219707 %
Fold 8 : accuracy: 80.79625292740047 % , F1-score: 47.270306258322236 %
Fold 9 : accuracy: 76.66666666666667 % , F1-score: 41.1451398135819 %
Fold 10 : accuracy: 78.59078590785907 % , F1-score: 39.41411451398136 %


C = 0.01 , penalty = l2
Fold 1 : accuracy: 76.01010101010101 % , F1-score: 41.67776298268975 %
Fold 2 : accuracy: 80.10204081632652 % , F1-score: 43.94141145139814 %
Fold 3 : accuracy: 82.1608040201005 % , F1-score: 45.67243675099867 %
Fold 4 : accuracy: 79.89556135770235 % , F1-score: 42.47669773635153 %
Fold 5 : accuracy: 80.72916666666666 % , F1-score: 42.876165113182424 %
Fold 6 : accuracy: 80.72916666666666 % , F1-score: 43.275632490013315 %
Fold 7 : accuracy: 81.16343490304709 % , F1-score: 41.54460719041278 %
Fold 8 : accuracy: 80.70588235294117 % , F1-score: 47.4034620505992 %
Fold 9 : accuracy: 76.60668380462725 % , F1-score: 41.01198402130493 %
Fold 10 : accuracy: 78.59078590785907 % , F1-score: 40.21304926764314 %


C = 0.1 , penalty = l1
Fold 1 : accuracy: 75.78125 % , F1-score: 42.47669773635153 %
Fold 2 : accuracy: 80.6282722513089 % , F1-score: 44.607190412782955 %
Fold 3 : accuracy: 82.29166666666666 % , F1-score: 46.870838881491345 %
Fold 4 : accuracy: 78.76344086021506 % , F1-score: 43.80825565912117 %
```

The accuracy across the folds throughout the grid search is fairly stable (revolves around 80%). I decided to use the micro F1-score metric to in choosing the most optimal hyperparameters. I know that due to income inequality the majority of survey respondents have a salary bucket which falls into one of the lower income categories, thus the data in this assignment most likely has unbalanced classes.

The reason I chose micro F1-score over accuracy is because micro F1-score provides a more accurate metric compared to accuracy when used on data with unbalanced classes.

The mean cross-validation accuracy, accuracy variance, and F1-score of this optimal model can be seen below:

```
{'C': 0.1, 'penalty': 'l2'}
best accuracy: 79.60893854748603 %, variance = 4.338371193524006 %
best F1 score: 44.51398135818908 %
The optimal log model uses C= 0.1 and penalty = l2
```

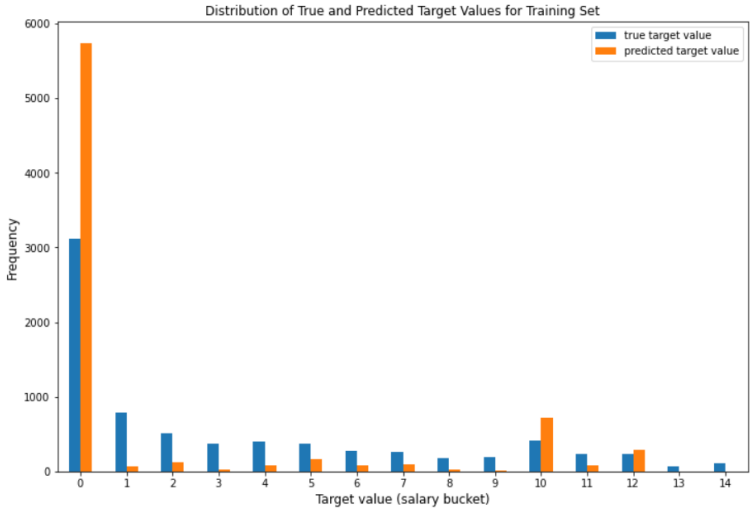
The optimal hyperparameters for the model is C = 0.1 and penalty = L2. When using the training data, the model has a cross-validation accuracy of 79.6% and F1-score of 44.5%. In terms of bias-variance tradeoff, this optimal model should have the smallest possible prediction error out of all the model variations that were tested in the grid search. In other words, this optimal model minimizes the bias^2() + var() term in the prediction error equation.

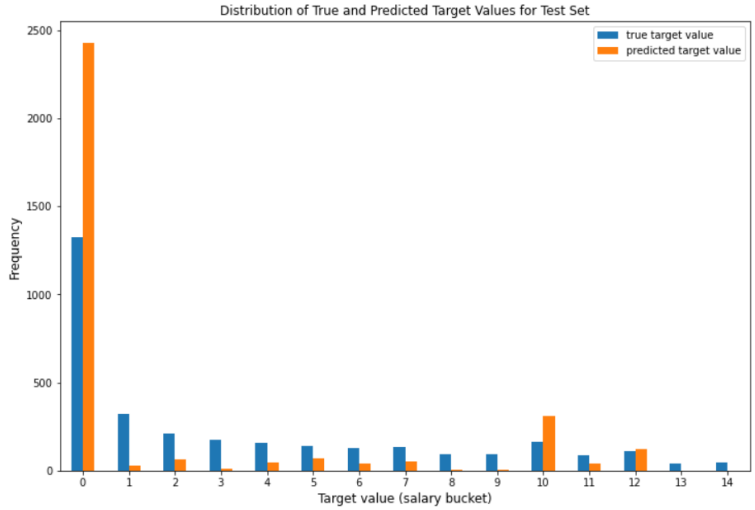
Model Performance on Test Data

Upon using the optimal model on the test data, I obtained the following metrics:

```
accuracy of test set: 80.68833652007649 %
F1-score of test set: 43.18111214662939 %
```

The model has a very similar performance on the test set compared to the training set. I believe that the overall fit of the model on the data is fairly good, since the accuracy on the test set is >80%. Similar to the training set, the F1-score on the test set is still under 50%, which I believe can be improved in some ways. Both the accuracy and the F1-score can be further improved by performing more complicated grid searches, such as introducing more hyperparameters in the grid search. My model might be slightly underfit, since the F1-score is fairly low (around 40% on the test set). In contrast, there is no issue with overfitting for my model since the metrics obtained from the training and test sets are almost the same. The distribution plots of true target values and predicted values on both sets of data can be seen below:





Discussion

Overall, I believe that the ordinal logistic regression model I have implemented performed fairly well in predicting the correct salary bucket of a survey respondent. After implementing the model and using it on the test set, I obtained the following insights:

- The country that a person works in (feature Q3 in the original data) greatly affects the annual income of the person.
- From the two distribution plots above, I was able to see that the model I implemented predicts the lower salary buckets very well, however struggles in correctly predicting some of the higher salary buckets correctly.