# Ordinal Logistic Regression Model Report

In this report, I will illustrate the optimal model I found for ordinal logistic regression on 'Kaggle ML/DS Survey' dataset in following parts, like how to choose model, how the model performed and other process when implementing model. The purpose of this report is to analyze the impact sector of yearly compensation, train and validate the multi-class ordinal model to predict the survey respondent's salary bucket.

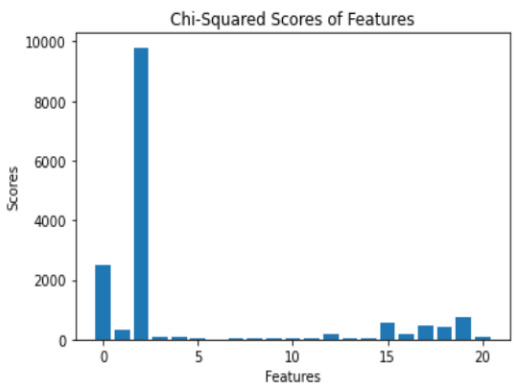
- Data Cleaning

From raw dataset, we have more than 300 questions in the survey, these questions considered as features. Firstly, we need to drop questions with majority of null answers. I set threshold of 9000 out of 15391 samples. Because most of these questions where null answers over 9000/15391 are part question, which were answered by small group of people. For example, in Q7, most respondent use "Python" answered in Q7_part1, a few people answered other programming languages in other parts. Thus, these part questions are not important to be covered in the model. I dropped other parts and left 1 part with majority of answers. Then I filled null values in this only left part as "other", for example in Q7, encoded them with "Python" as 1, "other" as 0. Similar method applied to other left questions. These answers do not carry ordinal sequence, so I used "LabelEncoder" to transfer all categorical answers to labels. In the end, I got a dataset with 21 features (questions) for each sample (survey respondent).

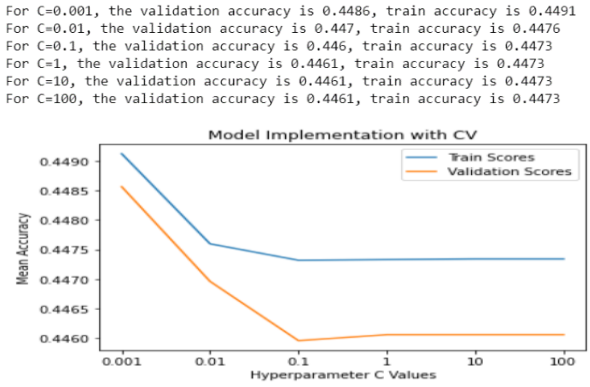
- Exploratory Data Analysis and Feature Selection

First, I split dataset to 70% training set and 30% test set, yearly compensation in Q25 as target values. The target value has 15 class range from 0 to 14, with an ordinal sequence. Class 0 is the lowest salary bucket, while class 14 is the highest one. Then I plot the dataset correlation heatmap (in appendix). Focusing on the Q25 correlation column, Q1 and Q3 with relatively light colors, which are age and country of origin, are correlated with Q25 compared to other questions. Q3 is the most related original attributes.

Then I applied feature selection in dataset. Given these 20 features, we still need process to determine the importance of these features, to what extend they can impact on the predictions of target value. Feature selection is the key process of feature engineering, which helps us improve the performance of the model, by giving importance order of these features. In this assignment, I chose Chi-Squared test. Because our dataset input and output are both categorical, Chi-Squared test is the most used feature selection algorithm for this type of dataset. The plot of Chi-Squared test scores (Figure 1) explicitly shows Q3 (feature 2) is the most correlated feature, and Q1 (feature 0) is the second most correlated. Then I dropped features with Chi-Squared test scores less than 100. Giving up these unimportant features could make the model less impacted by the noise. In the end, my final dataset has 10 features.




(Figure 1)




(Figure 2)

- Model Implementation

To implement ordinal regression by using “logistic_regression” model in scikit-learn, we need to transfer 15 ordinal target values into 15 binary classification cases to train the model. I defined a function named “ordinal_logit” to do so. The input of the function included “logistic_regression” model, training set for fit and validation set for model prediction. By using “predict_proba” to get probability for 0 and 1 in each binary classification case, the algorithm will predict the salary bucket of each sample to be with the highest probability for validation set. The output are probability data frame and the predicted class. Then I defined a “run_kfolds” function for 10-folds cross validation, the mean accuracy across 10 folds is 0.446 and variance is 0.0001.

I implement the model with only one hyperparameter C tuning, keep all other parameters as default. In figure2, we got the highest accuracy when c=0.001, and the accuracy in both training set and validation set almost the same. We know that low value of c represents the model put more weight on the complexity penalty at expense of fitting training set, which means the dataset is not as representative as standard real-world data. Though the training scores and validation scores are almost the same, I believe this model is still somewhat underfit due to the poor performance on dataset, as the high bias of the model.

Standardization is not necessary for this situation since I already encoded all categorical input with labels. Normalization does not improve the model.

- Model Tuning

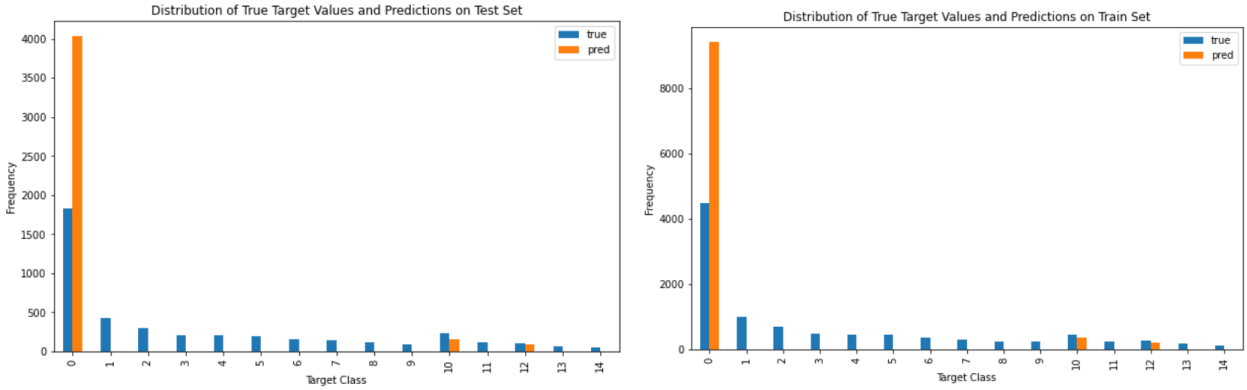
For model tuning, we need to identify all hyperparameters. As the scikit-learn logistic regression has been implemented, the hyperparameters are still penalty, solver, c values, multi-class, max-iteration and so on. I chose c values and solver for model tuning. I manually iterated all predetermined hyperparameters for grid search, and it returned the best parameters are c value of 0.001 and solver of ‘sag’. The standard deviation is 0.011, cross-validation score is 44.876%, which is higher than simple implemented model in part3. I set f1-scores as accuracy since the dataset is unbalanced because of the high proportion of class 0. F1-score gives a more

accurate metric than 'accuracy_score' given unbalanced dataset. As the optimal model with c=0.001 and solver='sag' has the accuracy, the prediction error should be the lowest. In the theory of bias-variance trade off, the optimal model is the best model which balanced the bias and variance at the feasible lowest point.

Given the predicted class from the optimal model, we can plot the feature importance using correlation matrix. From figure 3 in appendix, Q1(age) and Q3(country) are still importance, but the most importance feature becomes age, rather than country in the previous feature selection process. After dropping some unimportant features in the middle, all left features become more importance than before. The optimal model focusses more on the age of data scientists to predict their salary bucket, which makes sense. Older data scientists have more professional experience on ML/DS field and bounded with higher yearly compensation.

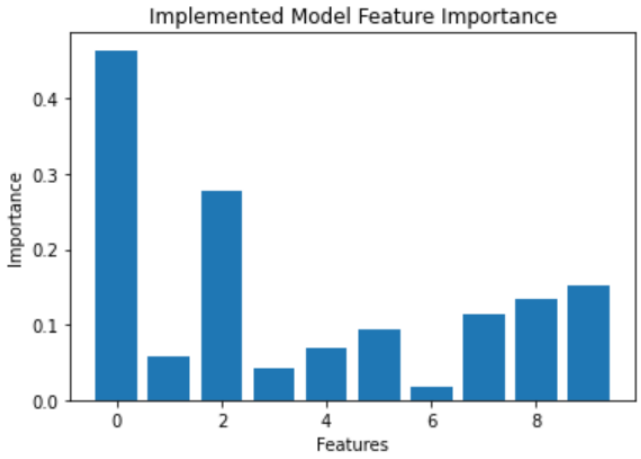
- Testing and Discussion

Applying the optimal model metric on the test set, I got accuracy of 42.994%. The performance on test set is consistent with previous training results and scores are similar. Because accuracy is still around 40%, my model is underfitted with poor performance on both training set and test set. We need to find some approach to improve the performance. For example, we can introduce more hyperparameters to tune the model. We can also put more effort in feature engineering. Due to the complexity of dataset, how to deal with massive features is the key point. I simply dropped majority of features with large proportion of null values, left only 1 part for some question. This method is limited, and we can improve more on how to choose importance features and encode them more appropriately.

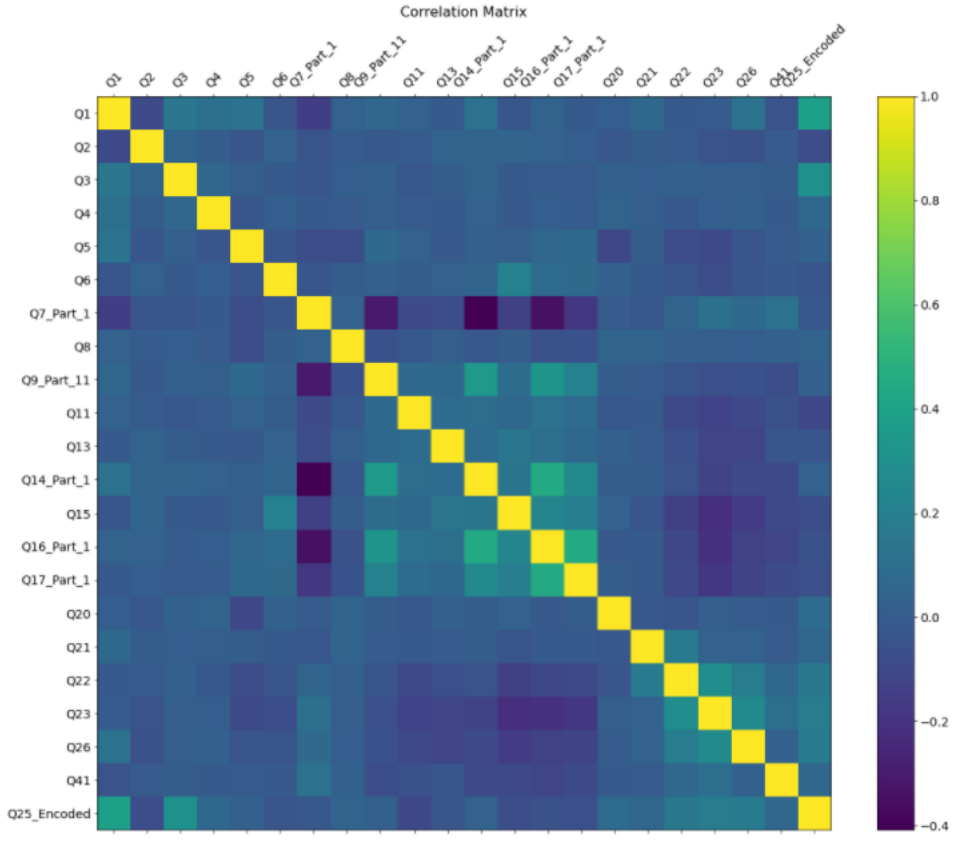



Given the plot above, which is the comparison of predicted and true salary buckets distributions on both training set and test set, we can see the model predicted three classes: bucket 0, 10 and 12. The model predicted large portion of class 0, which is also the majority class in true targets. The model performance on lower income samples is fairly good, but still need to improve on other small portion of classes which have not been predicted.

# Appendix



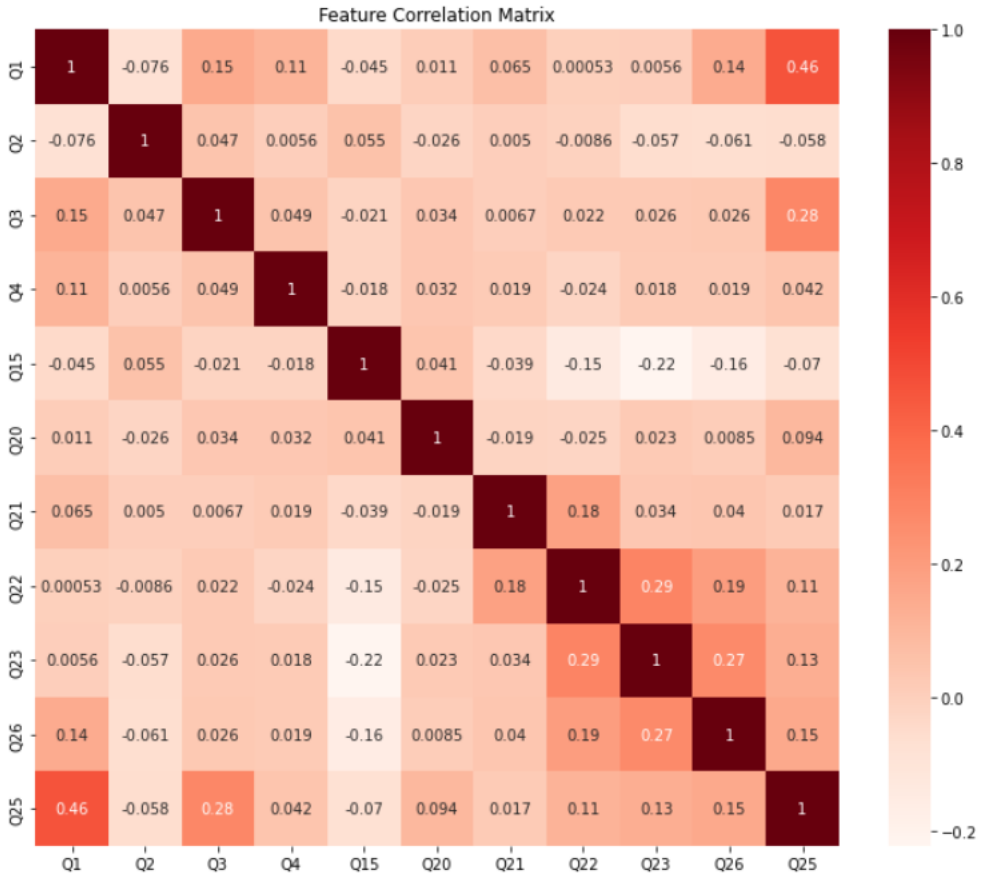

(Figure 3. Feature Importance after Implementation of Model)




(Figure 4. Correlation Matrix Heatmap on Feature Selection)

```
Fold 1 accuracy: 0.4348697394789579
Fold 2 accuracy: 0.46092184368737477
Fold 3 accuracy: 0.4438877755511022
Fold 4 accuracy: 0.4649298597194389
Fold 5 accuracy: 0.4483450351053159
Fold 6 accuracy: 0.436308926780341
Fold 7 accuracy: 0.4272818455366098
Fold 8 accuracy: 0.43931795386158473
Fold 9 accuracy: 0.45135406218655966
Fold 10 accuracy: 0.45336008024072216
Mean Accuracy: 0.44605771221480073
Variance: 0.00012865837923721168
```

(Figure 5. 10-Folds Cross Validation Accuracy Table)




(Figure 6. Correlation Matrix Heatmap on Optimal Model Performance)