MIE 1624: A2 Report

Date: 2023-03-12
Student: William Hazen
Student ID: 1009231225
Institution: University of Toronto
Faculty: Department of Mechanical & Industrial Engineering

Objective

The purpose of this assignment is to train, validate, and tune multi-class ordinal classification models that can predict a survey respondent's current yearly compensation bucket, based on a set of survey responses by a data scientist. Column Q29 "What is your current yearly compensation (approximate $USD)?" is the targeted column.

1 Data cleaning

Initially working with the '*clean_ kaggle_ data_* 2022.*csv*' I noticed that there were NaN values scattered through the 298 columns. So to avoid changing the target column Q29 and its variants (Q29 _ Encoded & Q29 _ buckets), I stored their values and then proceeded to drop them from the dataframe. To initially see the number of NaNs in each column, I found a large majority of them had high NaN values. According to statistical articles, it is common practice to treat data that has more than 40% missing variables to be considered as hypothesis generating. Therefore if a column has more than 40% missing values, I deemed that column to be dropped. This totaled 276 columns being dropped leaving 18 features left in the dataframe. Now with the remaining columns, I wanted to encode the features numerically, however I noticed that some features were categorical and some had values where encoding should include order. For example, Q2 (Age) values '18-21' and '22-24' should be 0 and 1 respectably rather then randomly assigning numerical values. So I created 2 separate dataframes where one is entirely categorical features and one the other is ordered base columns - the listed of ordered based columns were the following, 'Q2 (Age)', 'Q11 (Programming experience)', 'Q16 (machine learning experience)', 'Q25 (Size of company)', 'Q26 (Number of individuals responsible for data science workloads)', 'Q30 (Money spent on machine learning or cloud services).

To note, dealing with the NaN values for categorical columns was replaces with 'Unknown' due to the values being mainly strings, and for the ordered based columns list above, have numerical values thus NaN were replaced with the mode of the column (most common value). After dealing with the NaNs I then converted categorical features to indicator variables (using pandas.get _ dummies) resulting in a one hot encoded row for each survey respondent. Lastly, before concatenating both dataframes, I filtered any columns that had were more than 95% empty - this removed countries (Q4) that had a very small selection - as well as columns that 'Other' and 'unknown' in the end of their names. The reasoning was the values were vague and may be difficult to interpret when one hot encoded. The end results in an encoded cleaned dataframe show in Figure 1.

2 Exploratory data analysis and feature selection

2.1 Exploratory Data Analysis

To start, the exploratory data analysis (EDA) process, a correlation plot was created with the Q29 _ Encoded target column. The result is shown in Figure 2, the most correlated column was Q4 _ USA which had a 0.584 correlation value and the least correlated was Q4 _ India. This is interesting insight as from assignment 1, India and USA were the first and second most common countries selected in the survey response such that they have an opposing correlation with the yearly compensation column. Unsurprisingly, Q11 and Q16 had a high correlation with Q29 as experience with programming and machine learning is a monetizable skill in the field of data science. Furthermore, I conducted a Chi^2 test where Q13 _ 11 _ Jupyter Notebook had a high p-value with Q29 so this indicates the null hypothesis can be rejected, likewise, Q11, Q16, Q4 _ USA had a p-value of 0 which reinforces the results we got in the correction matrix highlighting the importance of these three features.

2.2 Feature Selection

Feature engineering was applied in order to select appropriate features. It is important to apply feature engineering since the original dataset started with 298 columns, to which if a model was fitted without feature engineering it would increase the chance of overfitting and slower computation time. Applying Lasso Regularization with hyperparameter tuning for alpha on the clean dataframe (Figure 1) resulted in 13 features that had coefficients greater than 0 while having 28 less than or equal to 0. From here, I dropped the 28 features from the clean dataframe resulting in a new dataframe with 13 important features shown in Figure 3 and 4.

3 Model implementation

The creation of the ordinal logistic regression model was made using the Sklean LogisticRegression function that would be looped over a range from 0 to 1 less the number of classes y _ train (Q29 _ Encoded). This allowed the LogisticRegression to become a multi-binary classification model created by training the LogisticRegression model to distinguish between the current class and all the previous classes. This is done by the cumulative probability of ordinal results of y _ train being less than a class 0...14 (low to high) such that from here the new probabilities are then subtracted by the previous probabilities so that each class can be calculated. I then selected the maximum probability out of all the classes for observation which allows the ordinal logistic regression (OLR) to be a multi-classification model. To note, I also store and average the coefficients of the logistic regression model that will be used in part 4. After 10-fold cross-validations, the average accuracy score was 41.363% with a standard deviation of 1.912%. From here, the LogisticRegression model from sklearn has an inverse of regularization strength hyperparameter 'C' such that after manually looping through variance values for C, the best parameter was chosen based on the best accuracy score which was C=0.1. During this loop, I was also able to calculate the bias, variance and mean squared error for that changed with C, such that Figure 5 shows the bias-variance tradeoff where we see as the C _ value increases the bias decreases and the variance increases, thus the most optimal c according to Figure 5 should be ≈ 0.025.

4 Model tuning

4.1 Hyperparameter Tuning

The following hyperparameters for the LogisticRegression sklearn model were, 'penalty', 'dual', 'tol', 'C', 'fit_ intercept', 'intercept _ scaling', 'class_ weight', 'random_ state', 'solver', 'max_ iter', 'multi_ class', 'verbose', 'warm_ start', 'n_ jobs', 'l1_ ratio'. Choosing C allowed the regularization strength to be controlled resulting in a model that can be appropriately fitted to the data, while choosing the Solver can help with the format of our dataset and goal. The 'liblinear' is limited to one-versus-rest schemes such that after performing a grid search with different C values and solvers, the best parameters were C = 100 and solver = liblinear which had a F1 score of 0.124.

4.2 Performance measures

To measure our model's performance, I use the F1-score as a metric to compare as the accuracy can may not be suitable for ordinal logistic regression problems, since it assumes that all misclassifications are equally important. Misclassification in an ordinal logistic regression problem may have different importance depending on the aspect of the problem. Therefore, F1-score would be a better metric as it is the harmonic mean of precision and recall. Now by using the best hyperparameters, the coefficients from the ordinal logistic regression after 10 folds was average to produce a feature importance plot (Figure 6) where there is notable differences from Figure 4 being the feature importance based on correlation with Q29_ Encoded. Q4 _ USA is still the most important feature, but the rest has been reevaluated - surprisingly Q4_ accounting/finance which was rated 13th in figure 4 is now created 4th from the ORL model.

5 Testing & Discussion

For testing, using the best hyperparameters mention above on the test dataset and the results produces had an F1 score of 0.108 and an accuracy of 40.06%. This was similar to the results of the training dataset such that when plotting a histogram for the classification for each class of Q29 of both the train and test datasets, the results show a general underfit. Figure 7 shows the histogram for the training dataset and its predictions, and Figure 8 shows the same plot but with the test dataset. Focusing on Figure 8, looking at the density lines of the histogram, the orange line (predictions) is failing to capture the aspects of the blue line (true labels). It appears to be over-predicting class 0 (0-$999), this is a result of an unbalanced dataset as class 0 has over 3000 instances whereas the other classes have around 300-500. The unblanced dataset could be the cause of the underfitting as hacing unevenly distributed classes makes it more difficult for our ORL model to capture the underlying patterns in the data. Overall, the objective for this assignment was to create an ordinal classification model that could predict a survey respondent's yearly compensation based on their answers. Though a model was created, the model's performance was poor and unable to accurately predict the correct classifications due to an unbalanced dataset. The lack of balanced representation of classes in the dataset hindered the model's ability to learn and make accurate predictions, especially for the minority classes.

Appendix

	Q2	Q11	Q16	Q25	Q26	Q30	Q7_2_Online courses (Coursera, EdX, etc)	Q7_2_unknown	Q12_1_Python	Q12_1_unknown	...
1	8.0	5.0	2.0	0.0	1.0	3.0	1	0	1	0	...
2	3.0	5.0	5.0	2.0	6.0	0.0	1	0	1	0	...
3	10.0	6.0	6.0	3.0	6.0	2.0	0	1	1	0	...
4	5.0	5.0	6.0	3.0	6.0	2.0	1	0	1	0	...
5	5.0	5.0	6.0	3.0	2.0	5.0	0	1	1	0	...
...	...	...	...	...	...	...	...	...	...	...	...
8132	5.0	5.0	1.0	4.0	6.0	0.0	1	0	1	0	...
8133	4.0	2.0	2.0	4.0	1.0	2.0	1	0	1	0	...
8134	5.0	2.0	0.0	0.0	0.0	1.0	1	0	1	0	...
8135	2.0	1.0	1.0	4.0	0.0	2.0	1	0	1	0	...
8136	4.0	3.0	1.0	0.0	0.0	0.0	0	1	1	0	...

8136 rows x 49 columns

Figure 1: Q1 Encoded Df

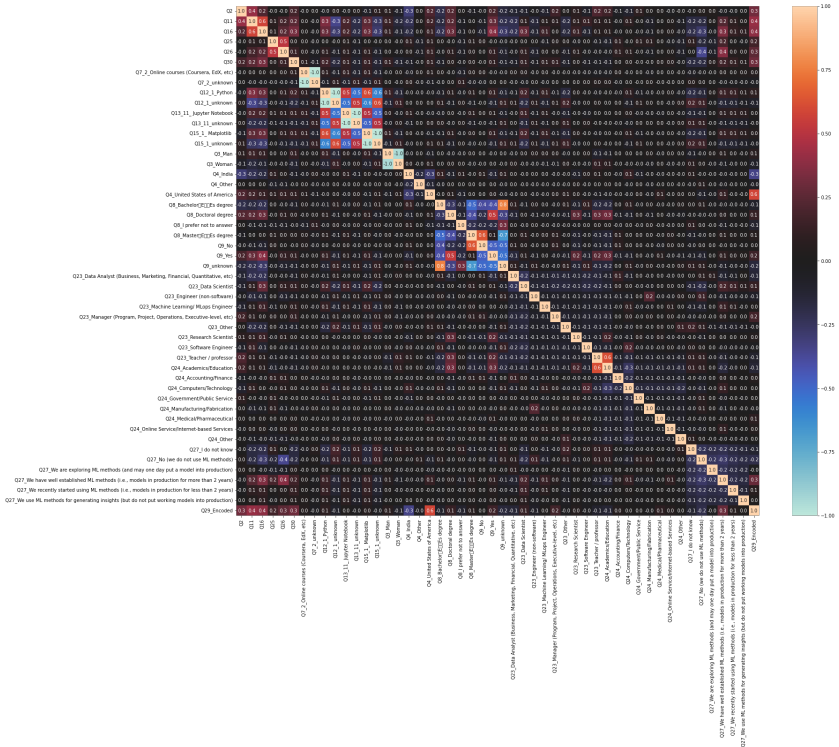

Figure 2: Q2 Correlation Plot

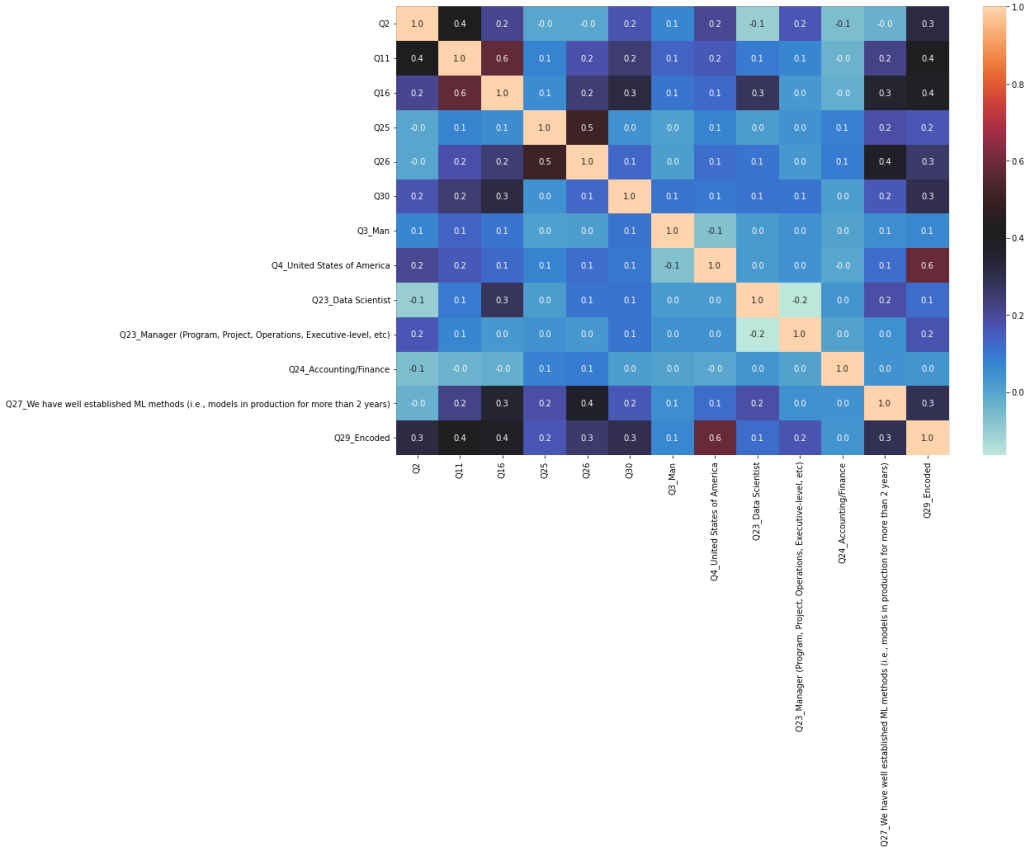

Figure 3: Q2 Good Features Correlation

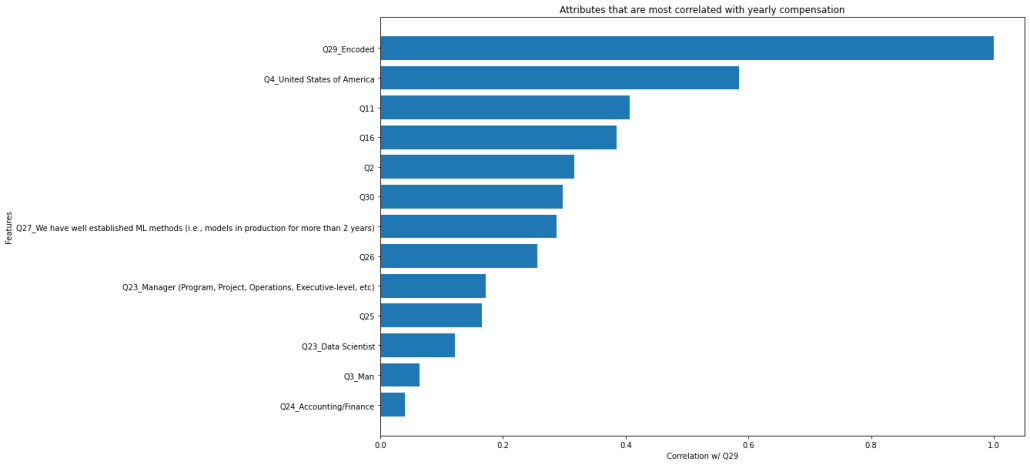

Figure 4: Q2 Feature Importance based on Correlation with Q29

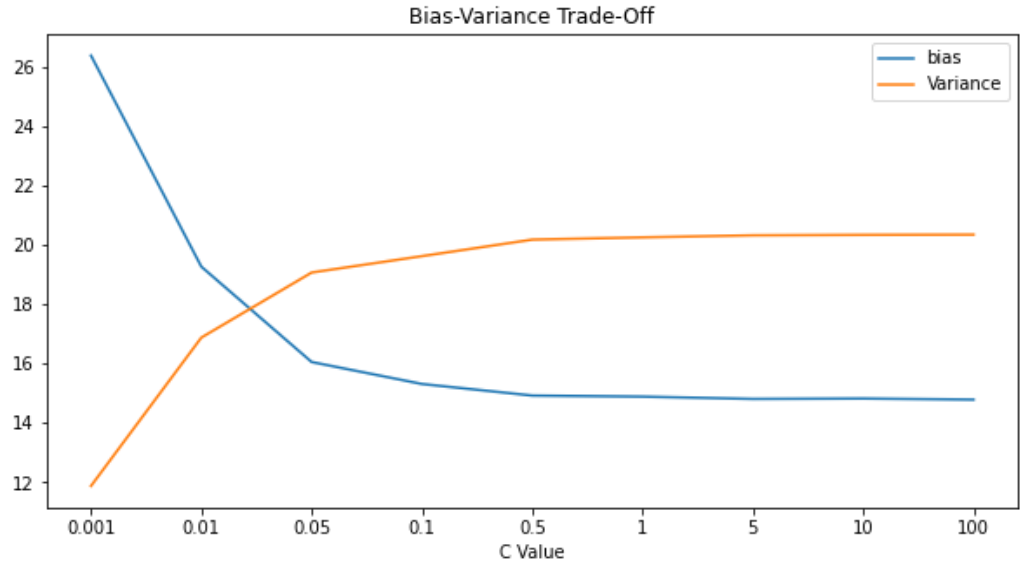

Figure 5: Q3 Bias Variance Tradeoff

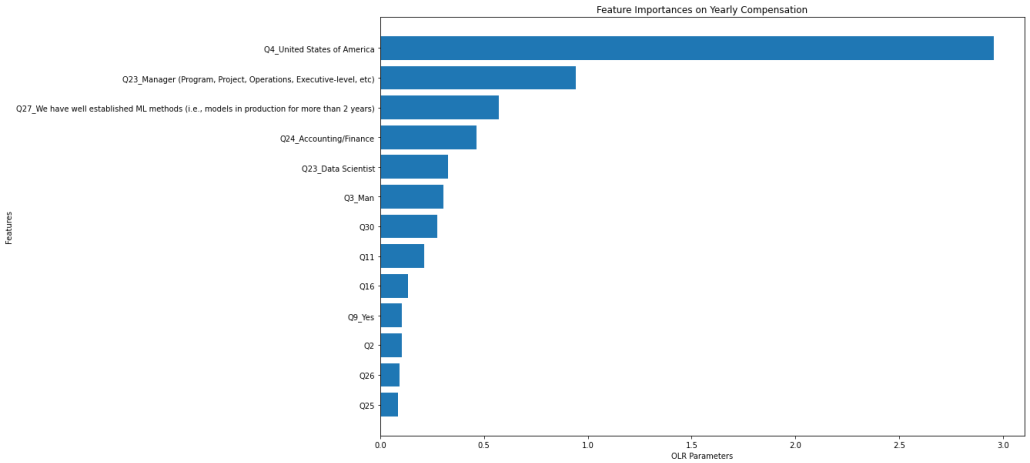

Figure 6: Q4 Feature Importance based on Ordinal Logistic model coefficient values

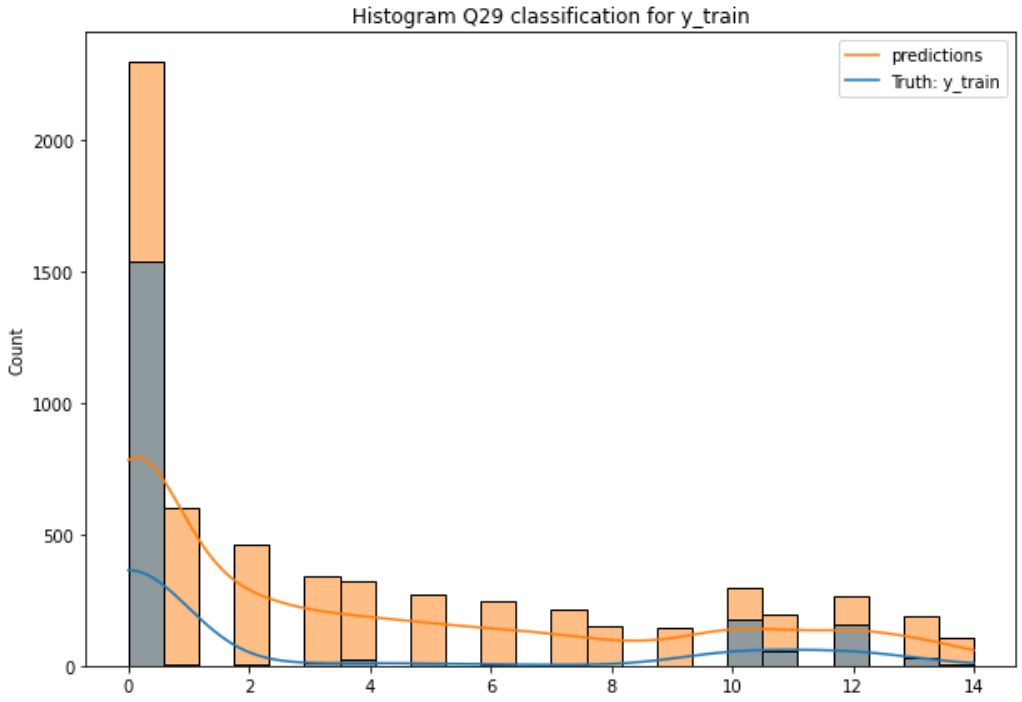

Figure 7: Q5 Train

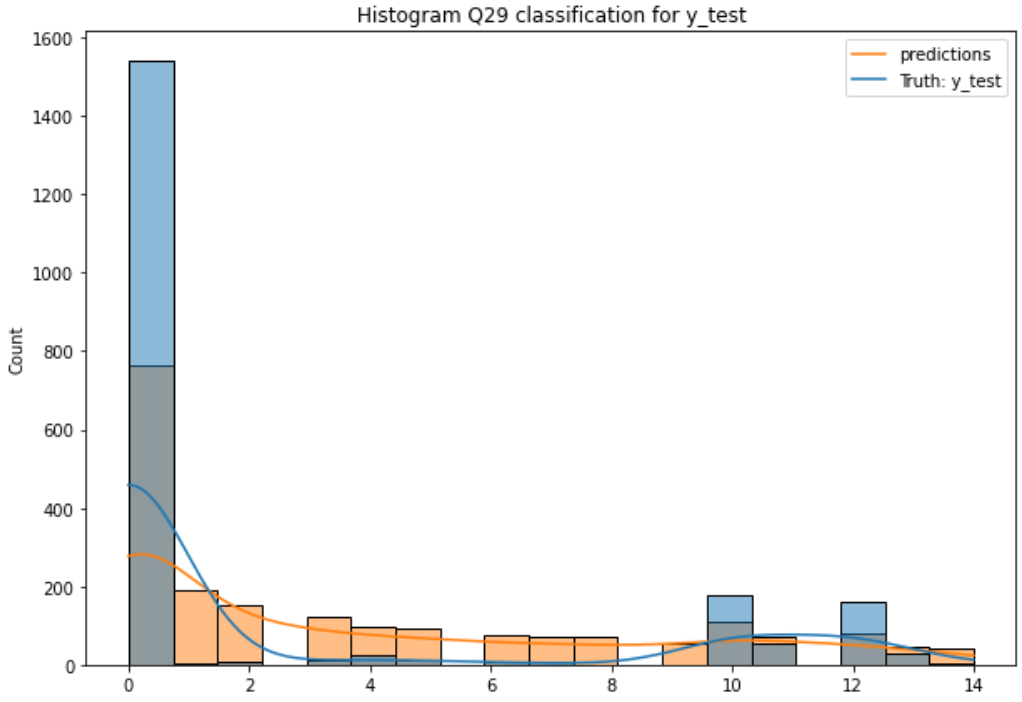

Figure 8: Q5 Test