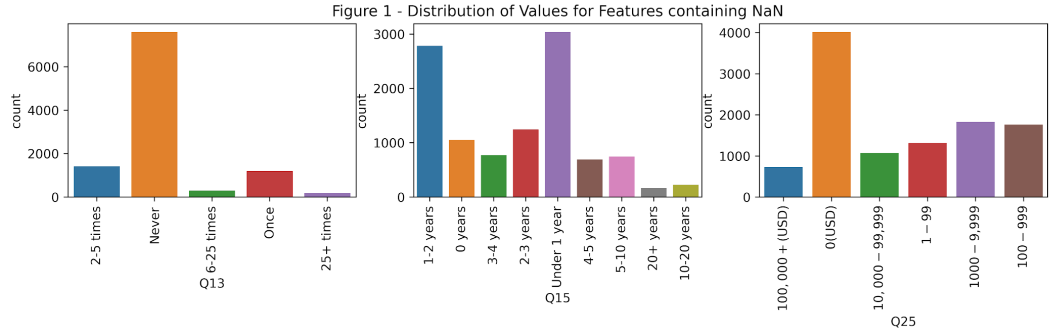
Sugumar Prabhakaran (**Id #:** 994126815) **Due Date:** 07 Mar 2021

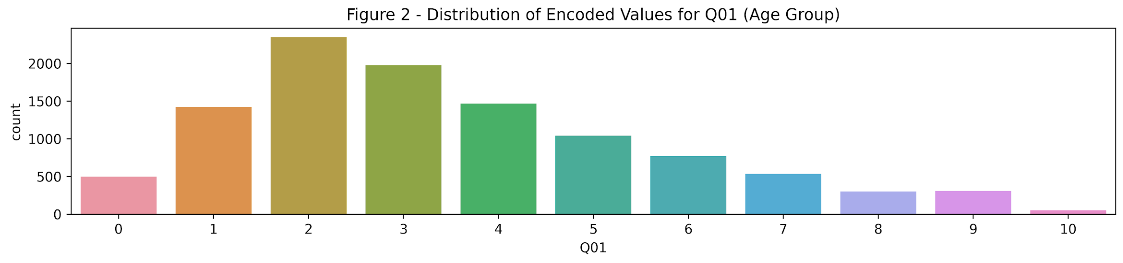
ASSIGNMENT 2 - MULTI-CLASS LOGISTICS REGRESSION CLASSIFIER

1. **Data Cleaning**. Dealing with missing (null) values and method for encoding categorical data was done according to three groups:

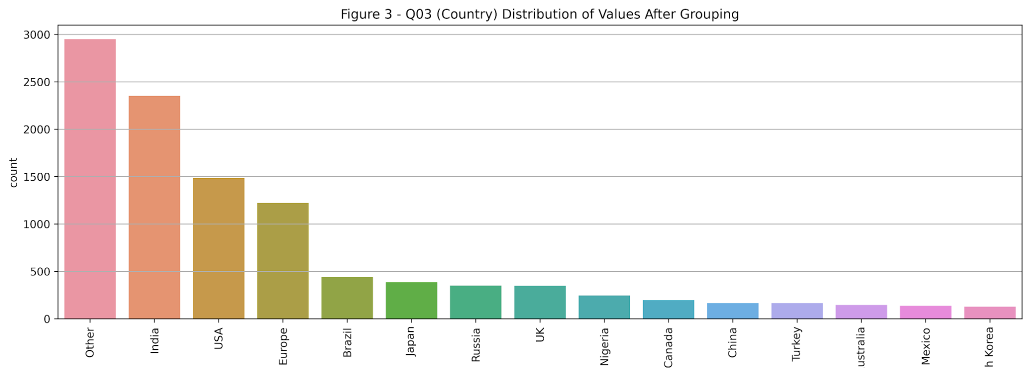
(a) Group 1: Single Column Questions with Ordered Categorical Data. There were 9 columns that fit this description: Q01, Q04, Q06, Q13, Q15, Q20, Q21, Q22 and Q25. First, null values were dealt with; only features Q13, Q15 and Q25 contained null values. From the distribution of values of these features (Figure 1), we replaced null values with the median value for each column.



Next, ordered values for each column were replaced by corresponding integers. An example of this is shown for Q01 (Age) below:



(b) Group 2: Single Column Questions with Unordered Categorical Data. There were 7 columns that fit this description: Q02 (*Gender*), Q03 (*Country*), Q05 (*Job Title*), Q11 (*Computing Platform*), Q30 (*Big Data Products Used*), Q32 (*BI Tools Used*) and Q38 (*Primary Tool*). Q08 (*Recommended programming language*) also met the description but was dropped since Q07 already contains a more direct version of the same question: "*What programming language do you use regularly?*" After looking at null value data for each of these columns, Q30 and Q32 were dropped since they contained less than 35% non-null values. Column Q11 and Q38 were missing less than 15% values so these were replaced using the median again, similarly to part (a). I grouped some values together for columns Q02 (ie. non-binary genders) and Q03 (originally over 55 countries) to reduce the overall number of features. Figure 3 shows the distribution of countries after grouping *'Europe'* countries together and countries not in the top 15 with *'Other'*. Finally, all the remaining group 2 columns: Q02, Q03, Q11, Q15 and Q38, were encoded using one-hot encoding.

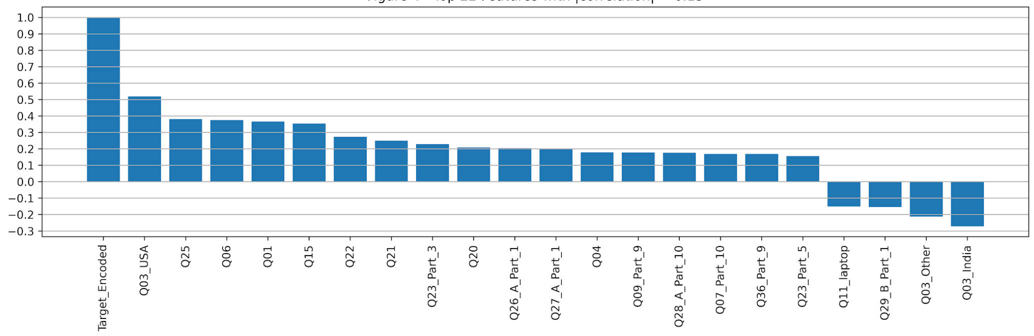


(c) Group 3: Multi-Column Questions. All the remaining columns (approx. 340) were already semi-encoded in that they were separated into columns for each possible survey response. For these, a loop was used to convert all null values to 0 and all non-null values to 1, effectively achieving one-hot encoding. This assumption was made since a lack of response in a column meant that the respondent didn't select that option and the only non-null value in each of these cells matched the survey response. This concluded our data cleaning and all encoded columns were saved in a new dataframe (df2).

2. Exploratory Data Analysis and Feature Selection

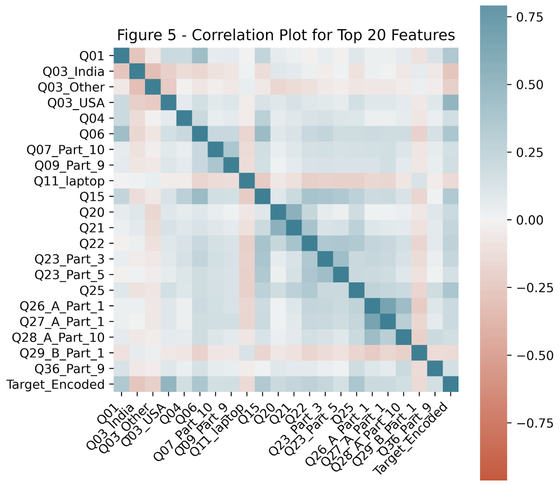
(a) Feature Importance. Using the pandas correlation method, the correlation of each feature with the target variable (renamed '*Target_Encoded*') was calculated. Figure 4 displays the top 22 features that have a correlation above 0.15 or below -0.15.

Figure 4 - Top 22 Features with |correlation| > 0.15



From this plot, we can see that many of the highly correlated-with-salary features are ones we would expect: Q03 (*United States*), Q06 (*Education Level*), Q01 (*Age*) and Q15 (*Yrs ML Experience*). In addition, Q03 (*India*) is quite negatively correlated with salary. Using these 22 features, we produce the correlation matrix plot in Figure 5 and can see that no features need to be dropped due to high correlation amongst each other. For example, we see that Q04 (*Age*), Q06 (*Education Level*) and Q15 (*Yrs ML Experience*) are correlated with each other but not excessively (ie. above 0.5).

Figure 5 - Correlation Plot for Top 20 Features

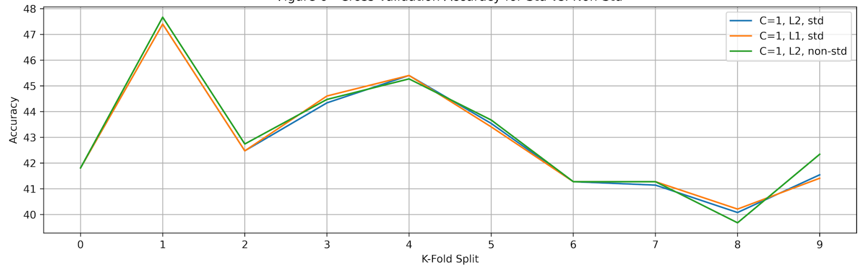


(b) Feature Engineering and Selection. Encoding and data cleaning, were elements of feature engineering, which is the process of transforming raw data into usable features. I used feature selection to simplify the model and speed up computations since we have 386 features. I selected a filter based method using correlation: only features with a correlation above 0.13 or below -0.13 were selected. As a result, only the top 31 most correlated features were selected.

3. Model Implementation

(a) 10-Fold Cross Validation. A function (**ordinal_log_regression**) was built to do ordinal multi-class logistic regression. A second function (**cross_validate_OLR**) was then built to implement the first function as part of k-fold cross validation. I executed the cross_validate_OLR three times to compare the effect of: (1) L2 vs. L1 regularization and (2) standardizing vs. not standardizing the data. Figure 6 depicts the performance of the three models over the 10 folds.

Figure 6 - Cross Validation Accuracy for Std vs. Non-Std



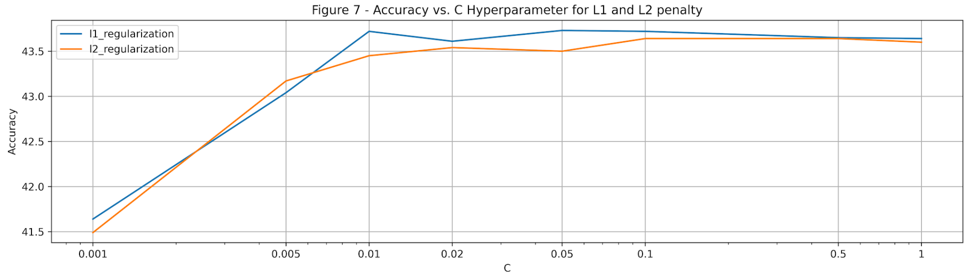
(b) Model Average Accuracy and Variance. All three models achieved an accuracy of about **43.6%** with a standard deviation of **2.2-2.3%**. Based on this, I conclude that for logistics regression, standardization does not make a significant difference.

4. Model Tuning

(a) Hyperparameter Tuning. The major hyperparameters that needed tuning were: the inverse regularization strength (C) and the penalty method (ie. L1 lasso or L2 ridge regularization). In addition, there were other potential hyperparameters such as the solver method (ie. newton-cg, etc.) but this did not affect any of the results so it was not tuned. Class_weight was experimented with since it is used for unbalanced data but this had a negative effect on my results so I removed it.

(b) Optimal Hyperparameters. Using a grid search-type algorithm, the following were determined as optimal hyperparameters: **C = 0.01** and **penalty = L1 regularization**. Accuracy scores for each model were within 3% of each other (see Figure 7). From the graph, we can see that all our models have a high bias so we selected the model with the lowest bias overall (highest accuracy). A detailed look at the cross-validation accuracies for each model (see notebook) show that all the models have a low variance as the results are quite similar across different training datasets.

Figure 7 - Accuracy vs. C Hyperparameter for L1 and L2 penalty



(c) Performance Metric. In order to compare different models for the hyper-tuning, **sklearn.metrics.accuracy_score** was used since its documentation states that the function can be used for multiclass classification as it computes subset accuracy.

5. Testing and Discussion

(a) Model Performance: Training Set vs. Test Set. The overall model performance was: **43.69%** for the **full training dataset** and **42.81%** on the **full test dataset**.

(b) Overall Model Fit. Our final model is underfitting the data since both our training and test accuracy are quite low. If we were overfitting the data, our training accuracy would at least be high. This model is a case of high bias and low variance. We have a low variance in that different datasets (ie. cross validation) produces very consistent and similar results. However, we have high bias, which indicates that the model is not complex enough and makes too many simplistic assumptions. I think the main issue is that the dataset is very imbalanced toward the lowest salary ($0-999) (see Figure 8 and 9). As a result, the model is predicting this class for too many cases. My research found that using weighted average as a metric or class_weights in logistics regression can help tune the model better. However, both of these were unsuccessful in producing better results and so I kept the original model. In order to improve accuracy, I would need to research methods to deal with imbalance such as oversampling, resampling and other classifiers.

(c) Distribution of True Target Variables vs. Predictions. Figures 8 and 9 compare the distribution of actual values with their predictions for both the training and test data sets. We can see that the main issue affecting the accuracy is that an excessive number of predictions are being classified in the $0-999 salary bucket. This is likely due to the imbalance of the original data set, which is skewed (see blue bars behind pink bars).

Figure 8 - Training Dataset Distribution

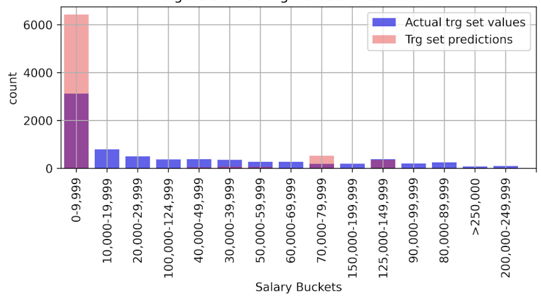
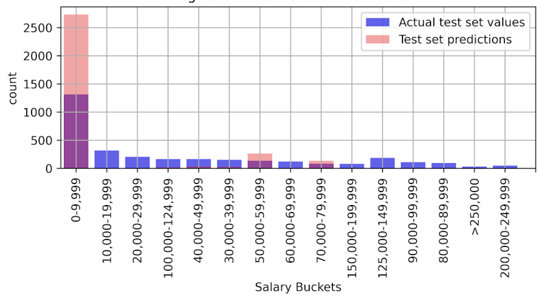


Figure 9 - Test Dataset Distribution


Actual test set values
Test set predictions
count
Salary Buckets