MIE 1624 Introduction to Data Science and Analytics – Winter 2023

Assignment 2

Name: Ayesha Patnaik, Student Number: 1008681696

The objective of this assignment is to 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..

Part-1, Data Cleaning :

In this data cleaning part, first I check for the total number of NaN values for each of the columns of dataframe.Furthermore, we identify the nature of the questions and categorize them. Each of these categories will be handled differently. The categories are as follows:

1. Categorical questions with order
2. Multiple choice questions
3. Miscellaneous(remaining) questions.

Next, observing from the dataframe, I see that there are columns from the MCQ questions that have specifically "Other" as the recorded response. So the order of data cleaning steps I performed are as follows:

1. I dropped the label 0 row as they contained questions which are not in our interest.
2. I check for the time taken by participants to do the survey. If for some participants the duration is less than a minute, I will not consider that response. This is because the information provided for 44 questions in less than a minute may not be thoughtful and can be considered unreliable. I found the minimum response was 134 seconds (more than 2 mins), so I kept all the participants. Then I deleted that column as it serves no further purpose to my model.
3. Then, I remove columns that are more than 80% empty as they have no significant information to contribute.
4. Next, I check for columns that contain only "Other" as responses and drop those columns. This is because I can't derive any meaningful conclusions from that column. I also drop "Q29" and "Q29_buckets" as we already have an encoded column "Q29_encoded" that will be our target column.
5. Next, I identify categorical questions with order. I check the distribution of the NaN values, confirm they are not heavily skewed, replace the NaN values with the mode of the respective column and do ordinal encoding.
6. Then I identify all the multiple-choice-questions. Since every choice of a MCQ question is listed as a column, only the choice selected is filled in the respective column for that particular participant. Rest all columns are NaN. So we will label the columns having non-null responses as 1 and the rest of the null columns as 0 for easier usage.
7. For the rest of the questions, I identify their potential as categorical data and impute the missing data with their mode values. Furthermore, I give categorical code to each response using *.cat.code.*
8. Finally, I drop the column "Q5" as it contains one type of response which is "0" implying none of the participants were students. My final dataset has 8136 rows and 74 columns.

Part-2, Exploratory Data Analysis:

In this assignment, we are going to use ordinal logistic regression using the cleaned data. However, even after cleaning, data contains some raw data that might be irrelevant, redundant, or noisy which in turn will adversely affect the performance of the machine learning model. Therefore, it is essential to perform feature

engineering to extract or select useful features from raw data. Objective is to improve the accuracy and efficiency of machine learning models. First, I studied the statistical summary of the data and then studied the target variable as seen in Figure-1a and Figure-1b. We inferred from the results that almost 37.58% of the data belongs to class 0 and the rest 63% is distributed among the other 14 classes. This makes it evident that the data is not balanced. To understand the relationship between features, I demonstrated *Pearson's correlation plot* as seen in the code file..The height of the bar indicates the absolute value of the correlation between that feature and the target variable. We then created a plot showing feature importances as shown in Figure-5. The height of the bar indicates the absolute value of the correlation between that feature and the target variable. The features are sorted in descending order of correlation strength, so the most important feature is at the top of the plot which is "Q4" with a correlation of almost 0.5, **indicating there is strongest correlation between the country of residence and annual salaries**. The four most important characteristics are "Country: Q4", "Years of experience in programming: Q11", "Years of experience in using ML methods : Q16" and "Age : Q2".Then I did feature selection by a *chi-squared independence test* as seen in Figure-2, where the null hypothesis is that two variables are independent and the alternate hypothesis assumes the opposite. I chose this technique as it deemed fit for the categorical dataset. Based on the test I listed a number of features ('Q9', 'Q13_5' ,'Q13_11' , 'Q15_2', 'Q35_1', 'Q44_4', 'Q7_5', 'Q42_8', 'Q12_1', 'Q7_2', 'Q6_7', 'Q15_1') that have p-values higher than 0.05 and hence they rejected the null hypothesis. These features were statistically insignificant and hence were dropped from the dataset.

Part-3, Model Implementation:

I performed binomial logistic regression 14 times to conduct the ordinal logistic regression for my target variable with 15 classes. In each binomial model, one class was considered as class 0 and the rest of the classes were considered as class 1. I did this in an increasing order of classes where the probabilities are calculated in a cumulative manner creating folds. Accuracies of each fold are calculated which are in similar ranges across the folds and the average accuracy for the folds is 34.803% with 2.952% standard deviation. Since we analyzed earlier that the data is unbalanced, we implement a “balanced” class_weightage in the binary logistic regression classifier. Treating the hyperparameter C (inverse of regularization strength) as a new model, I built and implemented models for various C values with a wide range. I calculated the accuracies for each model and plotted a bias-variance tradeoff for different C valued models or in other words for different levels of model complexities. The plot can be seen in Figure-3. As model complexity increases, bias decreases and variance increases. The optimal model is the point where bias and variance interact with each other which is around a model complexity of 1. Judging from the accuracy values, that exact best model seems to be the model with C = 0.01 with 39.66 % accuracy. I have also chosen to standardize the data before training the model based on the statistical summary obtained from the code captured in Figure-4. I inferred the wide gap in the means of the features and hence found it suitable to standardize my data to keep a uniform dataset for my model.

Part-4, Model Tuning:

Various hyperparameters that can be tuned in ordinal logistic regression model are penalty, dual, tol, C, fit_intercept, intercept_scaling, class_weight, random_state, solver, max_iter, multi_class, verbose, warm_start, n_jobs, and l1_ratio. Here, we selected to perform the tuning for two parameters which are : C and Solver. We did a gridsearch on various C values and solver choices with a balanced class_weight and found that for **C = 0.01** and a solver of **newton-cg** our model gives us the **best accuracy of 39.71% and F1 score of 0.39%**. Our assessment in the previous section is validated by the model tuning where we concluded our best C to be 0.01.

C is the inverse of regularization strength, which controls the amount of regularization applied to the model. A smaller C value will result in stronger regularization, and a larger value of C will result in weaker regularization. Regularization helps to prevent overfitting by penalizing large coefficients in the model. In other words, a lower value of C will lead to a simpler model, and a higher value of C will lead to a more complex model. In our case we have a relatively lower C value of 0.01

Solver determines the algorithm used in the optimization process to find the coefficients of the logistic regression model. The choice of solver depends on the size of the data set and the complexity of the model. We know that newton-cg is suitable for multi-class classification problems and hence it makes sense that after tuning we get best performance with newton-cg solver. It is a quasi-Newton method that uses a truncated Newton method to solve the optimization problem. It approximates the Hessian matrix using only first-order information and iteratively refines the solution until convergence.

Furthermore, in order to evaluate model performance we check both accuracy and F1-score. As was established before, our dataset is unbalanced. So even if the model predicts the right classification for the majority class resulting in higher accuracy, we would not know if it's performing well or not. Hence we need F1 scores to provide a better measure of model performance.

We then plotted another feature importance graph for our model to find out that the most important features to contribute in model predictions were Q28_5, Q16, Q17_6 and Q15_3 unlike in part-2 where the important features were Q4, Q11, Q16 and Q2. New feature importance plot can be seen in Figure-6.

Part-5, Testing and Discussion:

After testing our best model on the test set we get the following comparisons. Train set vs Test set:

For train set , accuracy = 39.71%

For test set, accuracy = 34.98% %

Since both the train accuracy and test accuracy are considerably low (below 40%) and also the test accuracy is less than the train accuracy, the **model is underfitting** with the dataset. **In order to improve accuracy of the model**, we already have done feature engineering and hyperparameter tuning of the model. So the next logical conclusion is to **get more data**. With more meaningful data, the accuracy can definitely improve. In addition we plotted the distribution of true target variable values and their predictions on both the training set and test set as seen in Figure-7 and Figure-8 (Appendix). If we look at the y-axis values for both plots, we see that even though the distribution between true values and prediction is more or less similar for training and test values, we analyzed that the prediction for the train set has higher values particularly for class 0, 4, 10 ans 12. While the prediction for test values are lower in value compared to train cases. There is significantly more weight in class 0 in both the predictions.

Overall the following are the insights:

1. The overall performance of the model is really poor with a 0.39% F1-score
2. The data is unbalanced and is not evenly distributed across the classes that significantly affects the performance of the model.
3. Significant removal of features during feature engineering and feature selection might be another cause of this uneven spread and hence underfit of the model.
4. Considerable resampling of data is required to increase the sample size so as to obtain more meaningful features to train the model for improved accuracies.

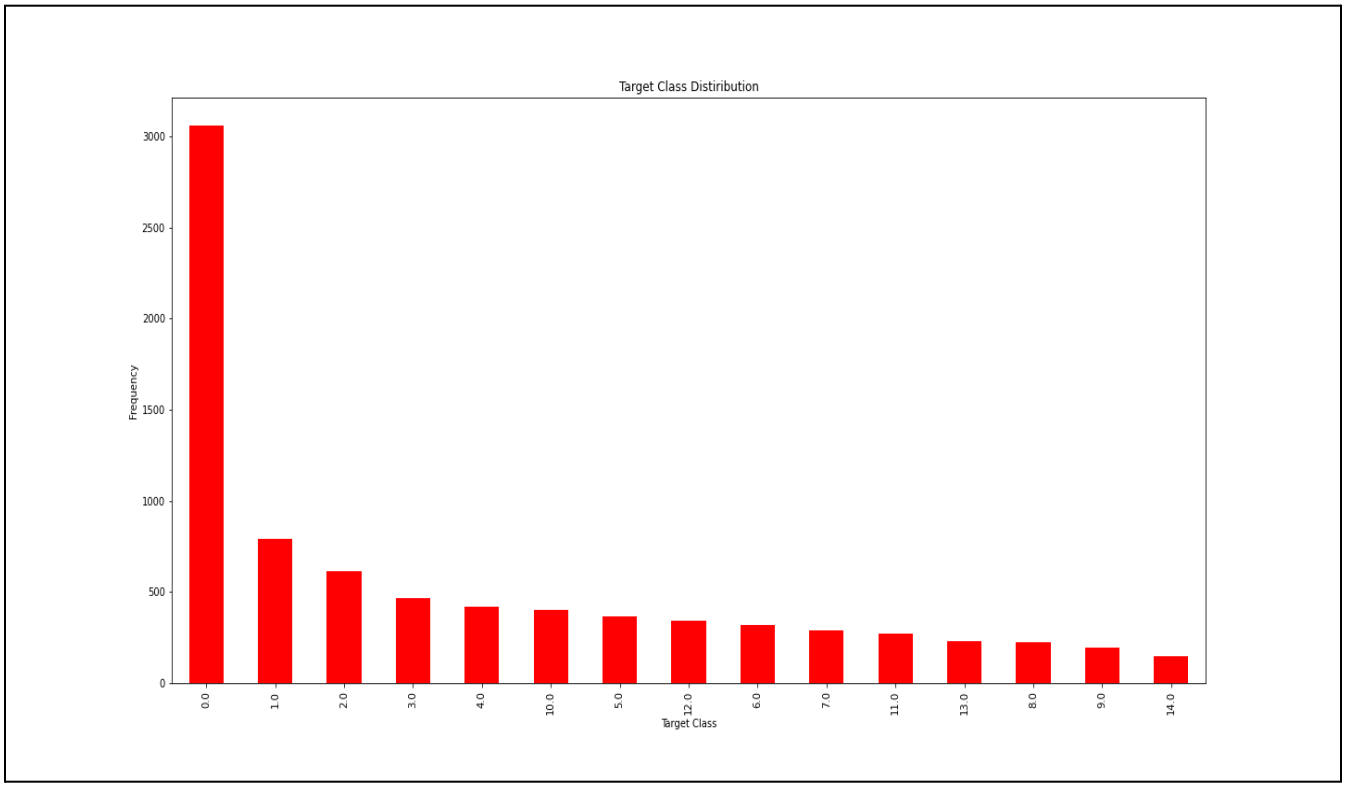
APPENDIX

1. Figure-1a: Target variable count

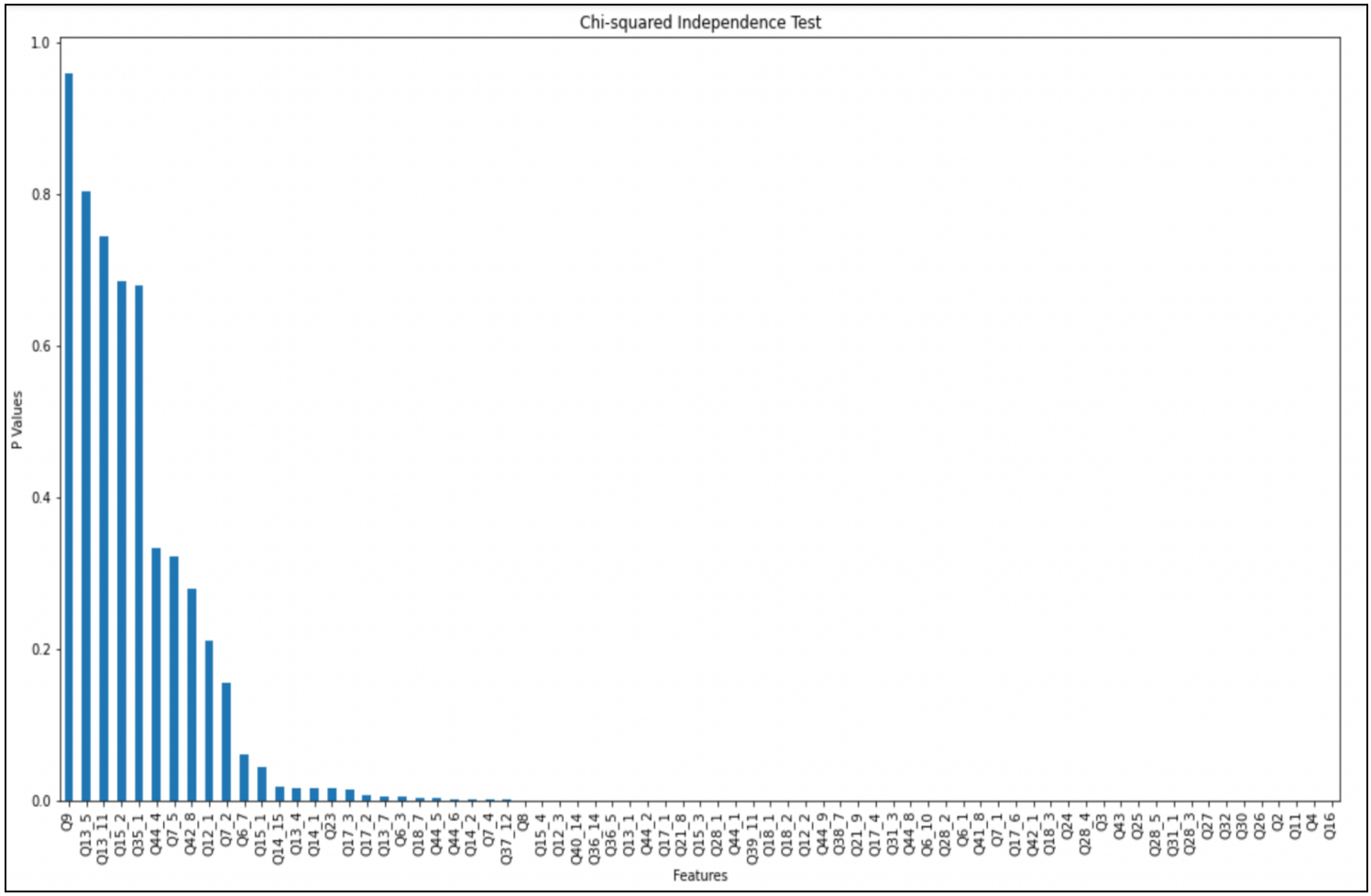
```
#Inspecting target variable
df["Q29_Encoded"].value_counts(normalize=True) #to see as percentage value
```

```
0.0     0.375860
1.0     0.097345
2.0     0.075467
3.0     0.057030
4.0     0.051745
10.0    0.049656
5.0     0.044985
12.0    0.042035
6.0     0.039086
7.0     0.035521
11.0    0.033063
13.0    0.028638
8.0     0.027286
9.0     0.024213
14.0    0.018068
Name: Q29_Encoded, dtype: float64
```

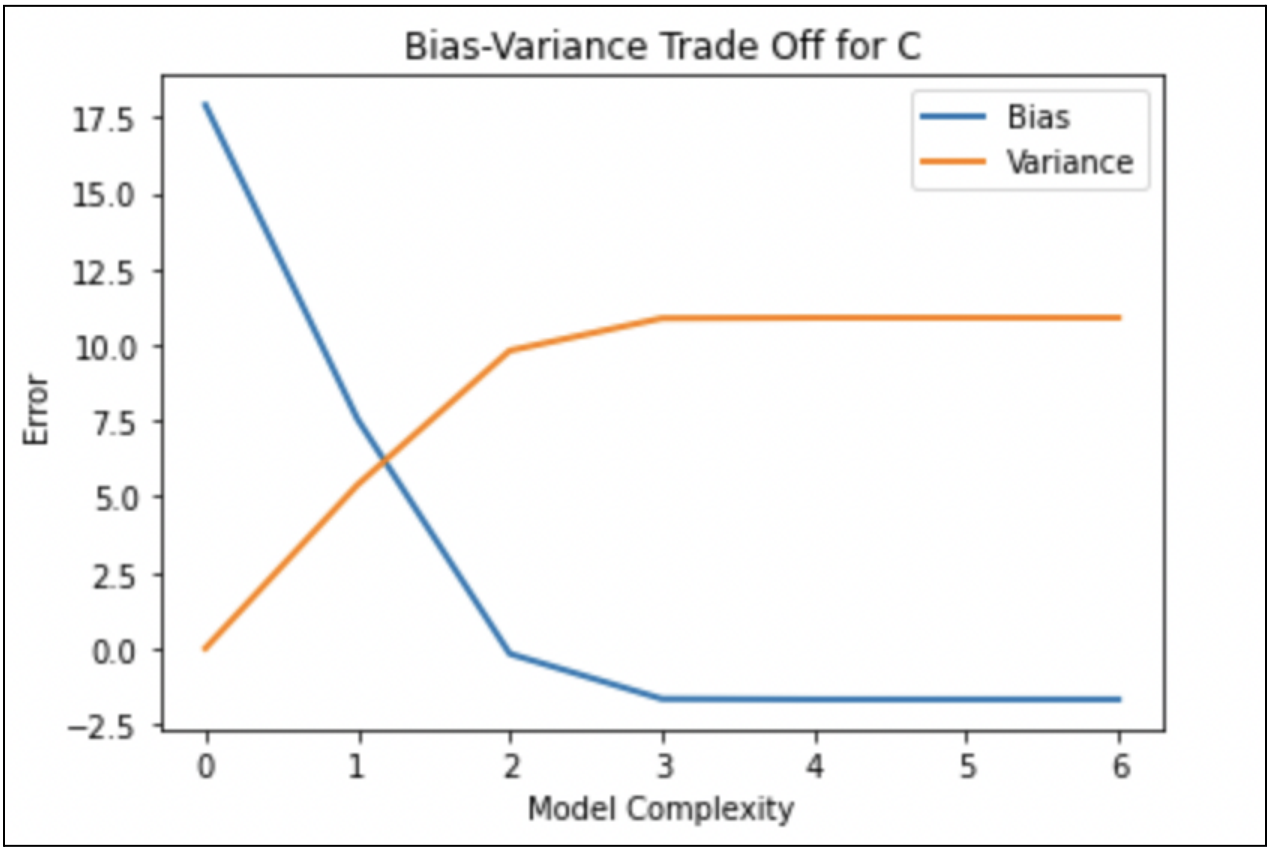
2. Figure-1b: Target Class Distribution



3. Figure-2: Chi-squared independence test



4. Figure-3: Bias-Variance Tradeoff

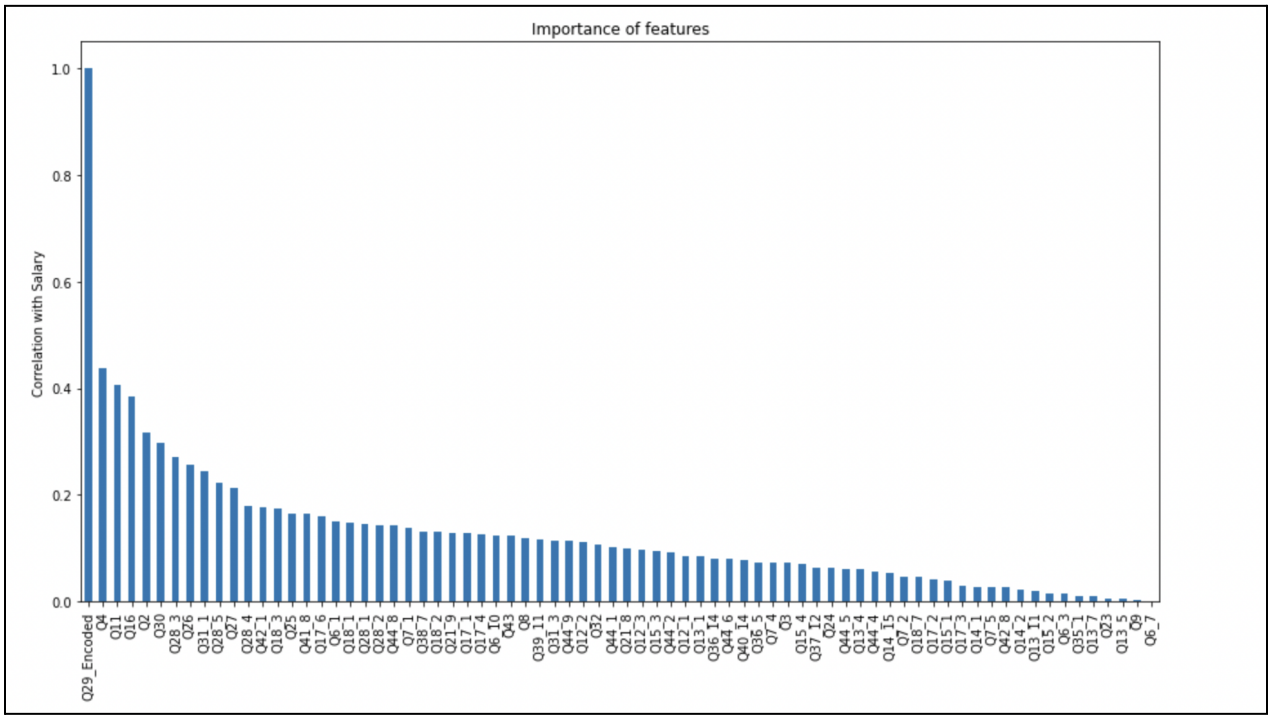


5. Figure-4: Statistical Summary of Data

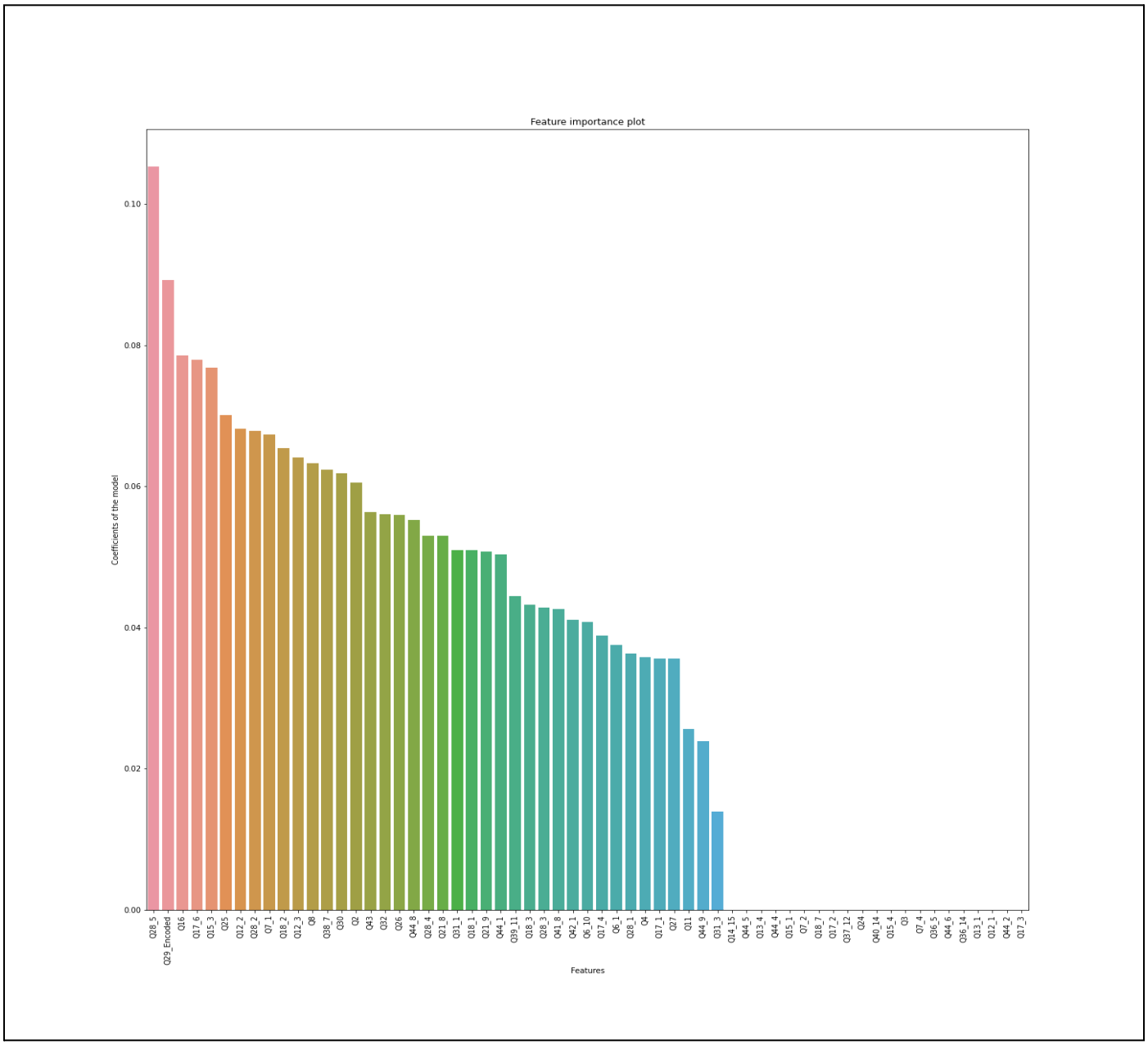
```
#summary of the dataframe
df.describe()
```

	Q2	Q3	Q4	Q8	Q9	Q11	Q16	Q23	Q24	Q25	Q26	Q27	Q30	
count	8136.000000	8136.000000	8136.000000	8136.000000	8136.000000	8136.000000	8136.000000	8136.000000	8136.000000	8136.000000	8136.000000	8136.000000	8136.000000	8136.0
mean	4.057276	0.706981	30.676254	2.837266	0.689897	3.125369	2.550885	6.551008	5.189405	2.048181	2.850910	2.260447	1.512414	1.9
std	2.254581	1.507834	16.382457	0.948880	0.462564	1.838539	2.088365	3.911951	4.313515	1.496282	2.203148	1.572470	1.361005	2.6
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	2.000000	0.000000	20.000000	2.000000	0.000000	2.000000	1.000000	4.000000	2.000000	1.000000	1.000000	1.000000	0.000000	1.0
50%	4.000000	0.000000	26.000000	3.000000	1.000000	3.000000	2.000000	7.000000	3.000000	2.000000	2.000000	2.000000	1.000000	1.0
75%	5.000000	0.000000	46.000000	3.000000	1.000000	5.000000	4.000000	10.000000	9.000000	3.000000	6.000000	3.000000	3.000000	1.0
max	10.000000	4.000000	57.000000	5.000000	1.000000	6.000000	7.000000	13.000000	14.000000	4.000000	6.000000	5.000000	5.000000	12.0

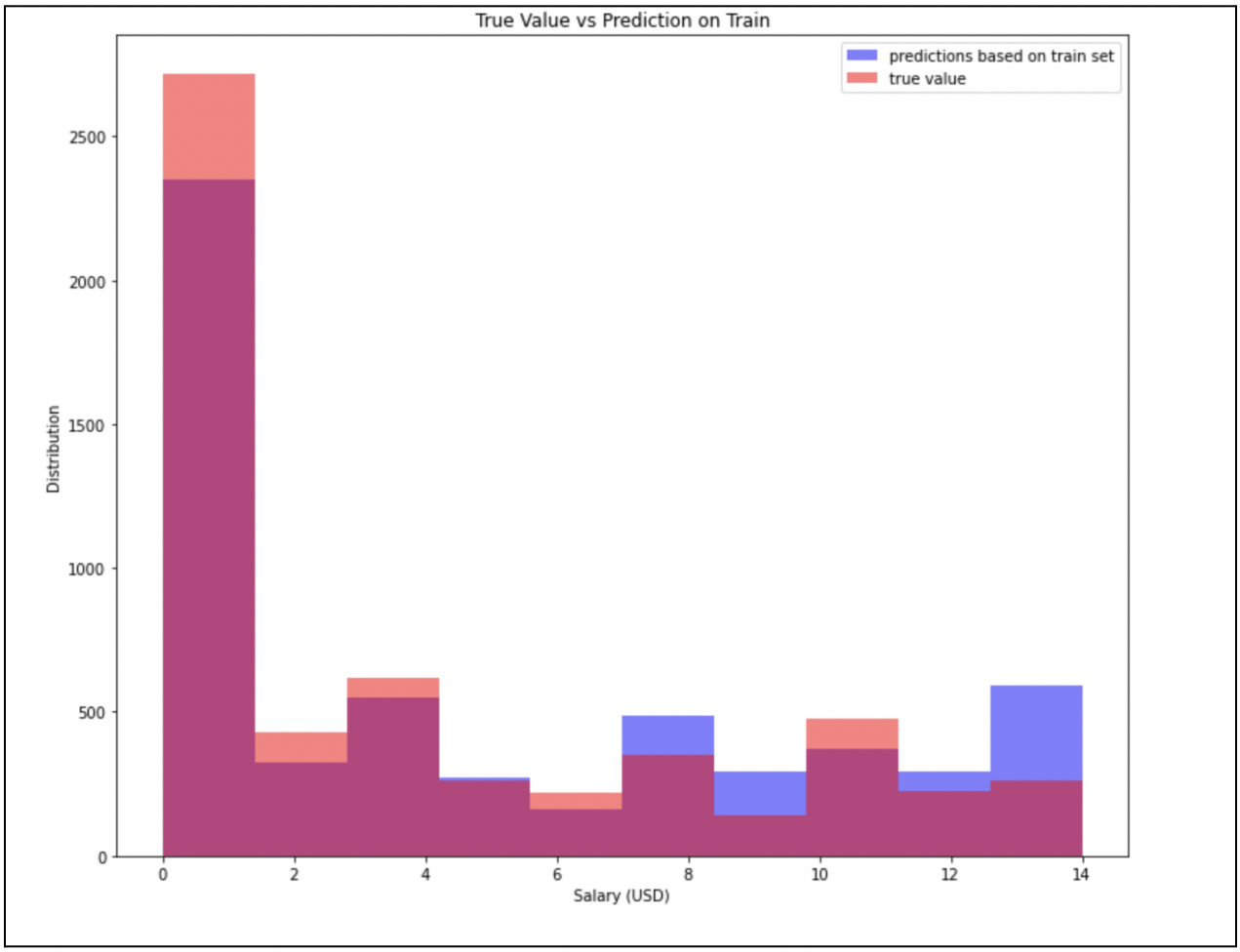
6. Figure-5: Feature Importances in Part-2



7. Figure-6: Feature Importances in Part-4



8. Figure-7:True value vs Prediction on Train



9. Figure-8: True value vs Prediction on Test