MIE1624 ANALYSIS REPORT

Assignment 2

Tongfei Li

1004759460 |

Data Cleaning

For multiple-choice questions, firstly, I go through all columns of the data frame and drop the columns for choice - other. The reason is that choose 'other' does not indicates the people has similarities among each other, the choice 'other' does not classify the person into any type. Secondly, I encode the remaining columns for multiple-choice as selected->1, not selected->0, to get the result data frame, I called build-in function 'pd.get_dummies'.
For the remaining data that has one question per column, I have basically four methods to clean.
The first method is to drop the whole column, this method is used when there are too many missing values (more than 1/3 of the whole dataset) or the variable is not related to response variable. I drop Q1 since time finishing a survey is not really related the person's salary, moreover, time is a very erratic variable, should be greatly affected by the environment and occasion the person open the survey. For example, the person can be disrupted halfway and come back hours later to finish the survey... Dropped Q5 since all values are the same, dropped Q9, Q22, Q32 and Q43 since more than 1/3 values are missing.
The second method is to drop rows(observations), it is performed along with the following 3rd and 4th methods. I used this method very carefully, to prevent lossing too many useful information. I only drop rows when there are very minor missing values (less than 5% of the whole dataset) like 'nan' and 'prefer not to say', the rational is that it may cause trouble if I categorize missing values into same class, since there's no evident that the persons are in same category.
The third method is to transform the values to ordinal categorical variable. More specifically, I assign value from low to high (0,1,2…) to represent increase level of values. I used this method on Q2 of age, Q8 of education, Q11&16 of experience, Q25 of company size and Q30 of expense.
The fourth method is to expand the single column to multiple columns, select the response of one choice into one column, and the value is 1 if the person chooses the choice, 0 of not, to get the result data frame, I also called build-in function 'pd.get_dummies'. This method is used if there's no ordinal characteristics among the choices, the resulted format is very similar to multiple-choice questions discussed in the last pasts. I also drop the column of choice 'other' and 'prefer not to say' when using this method, since there's no evident that the persons selected the choice 'other' are in same category. I used this method on Q3 of gender, Q23 of position and Q24 of industry.
Finally, I double checked and found that all missing values are handled, so data cleaning finished.

Exploratory data analysis and feature selection

Firstly, I wanted to generally see the strength of relation between features and target variable. So, I visualize feature importance based on correlation to response variable (Q29_Encoded). From the bar blot, we can observe some top features that affects salaries, for example Q11, Q16, Q2, which have more than 0.3 correlation with response variable.

Then I choose Random Forest for feature selection. I choose Random Forest because it is not a distance-based algorithm, so I don't need to standardize data in this part. Compared to the last approach, Random Forest takes in consideration of multicollinearity, while the features with highest relationship with response variable may still have high correlation between each other. Moreover, Random Forest is a stable algorithm when facing noises and outliers, it makes decision based on many decision trees. The second plot I draw is the feature importance based on random forest feature selection. We can see the top features are Q2, Q11, Q16, Q25, Q30 and Q8. Considered the dataset is very large, with 7000 observations and 14 levels of target salary, I choose top 80 important feature based on Random Forest output.

Model Implementation

I implemented ordinal logistic regression in a defined function, given training dataset and response variable X, y, the function slices the y from 0 to 13, fit logistic regression on modified y and training X set, and predict class for test data, at last, ordinal model output the class with highest probability based on logistic model. For ordinal models here, we want to standardize data before fitting in the model, since ordinal model is depended on logistic model, which is sensitive to distances.

Regarding 10-fold cross validation, the average accuracy is about 35.9% with standard deviation of 1.49%, among the 10 folds, the accuracy scores are very closed to each other, the lowest one is about 33.6% and highest about 38.4%. Therefore, our model accuracy for ordinal logistic regression is low but considerably stable.

Next, I tried 10 different models with different parameter C (regularization strength), the other hyperparameters are by default of Logistic Regression. I tried different C from 0.00001 to 100 and visualize their accuracy as well as their bias and variance. From the plot, we can see that, the maximum accuracy is reached at about C=0.01, using accuracy as performance measurement, the model with 0.01 regularization strength reaches best performance. This result can also be explained by bias-variance tradeoff, I plot bias and variance in the same figure for 10 models to visualize the dynamic changes in the variables as C increases. From left to right, C increases, and lower the penalty to parameter values, resulted in a more complex model, during the process, underfitting is reduced, afterward, overfitting is increased, so the variance increases and bias decreases. From C = 0.00001 to about 0.01, model complexity increases, bias decreasing and variance increasing, the trend level off at about 0.01. Since we don't want the model to be too simple to capture enough information, and we don't want it to be too complex, our best model among these ten can be found around C=0.01. This result is consistent with our best model chosen by accuracy score.

Model tunning

First, I calculate the proportion of each class among the whole response y. The dataset is imbalanced, with response class 0 takes up about 34% of total response. Thus, accuracy that only focuses on True Positives and True Negatives cannot be a good performance metric, since the model can get a high precision by producing the majority classes. A more rigorous metric can be F1 that also want to minimize False Negatives and False Positives.

For the model optimization part, I use grid search with sklearn package, and select C and solver for hyperparameter tunning. I calculate F1 macro average value of each fold and get cross validation F1 for each combination of hyperparameter. The final best model is the combination of 'liblinear' solver and C=0.9, with a highest cross validation F1 of 10.59%.
The feature importance of ordinal regression model selects = Q24, Q11, Q13_7, Q30 and Q25 as top 5 determining features, but only Q11 and Q25 are also top features selected by Random Forest, the other three are in the middle or nearly the least important feature selected by Random Forest. We can see that there is much difference in their selection, only some of the top features are the same. Additionally, Random Forest tends to have 6 'favorite' features, which have much higher importance than the others, the remaining 74 features shares nearly the same importance level, the trend of decrease level off from the 7^{th} value. While ordinal logistic is assigning the features gradually decreasing values in importance, we can see in the graph that the values keep decreasing all the time.

Model testing & discussion

The best model get has the combination of C=0.9 & solver=liblinear, I fit the model as model_LR_best, trained the model with scaled training data and get predictions from training and testing data respectively. I calculated F1 and accuracy score of training and testing data, F1 score is 16.96% and 10.47%, accuracy is 39.6% and 35.37% for training and testing dataset. For our best model, accuracy and F1 score is still very low, training scores are both a little bit higher than testing scores, but the performance is not good enough.
I have been doing profound reflection on my model's bad performance, and here are some possible reason and methodology to improve: Firstly, the data is very imbalanced, with class 0 has 34% of responses, it is possible that we should separate few classes under class 0, for example(0-2000, 2000-5000, 5000-9999) and reduce classes among class 1-14, since the observations in higher classes are not enough, if we modify our survey, and resulted in a more balanced dataset, the model should perform better. Secondly, we may try different models, we can choose a different classification model such as random forest or KNN. Another possible approach is to modify the survey to ask the participants to enter their approximate income, and make the response variable continuous, we may fit the model with liner regression instead. Finally, it is possible that I made mistakes in dataset cleaning, I dropped the question for country since there were too many categories under it, but I finally realize that salaries differ a lot in high-income and low-income countries, so I should have categorized countries into several income level to fit into our model.
Firstly, there's no evidence of overfitting on this model. If the model is overfitted, as we continuously increase model complexity (c>0.9), training score should increase and testing score should decrease, but in the F1 vs C graph, both training and testing level off after 0.9. On the underhand, it is suspected that the model is underfitting, an underfitted model has low scores on both training and testing dataset, since both F1 and accuracy score are very low, (train: 17 vs 10.5, test: 39.6 vs 35.4), I would say that the model is underfitted.
The distribution of true and predicted target variable among classes for training and testing are very similar. The distribution of whole bar plot is right skewed. For the majority class 0, there are much more predictions than actual values, while for many of the minority classes, the

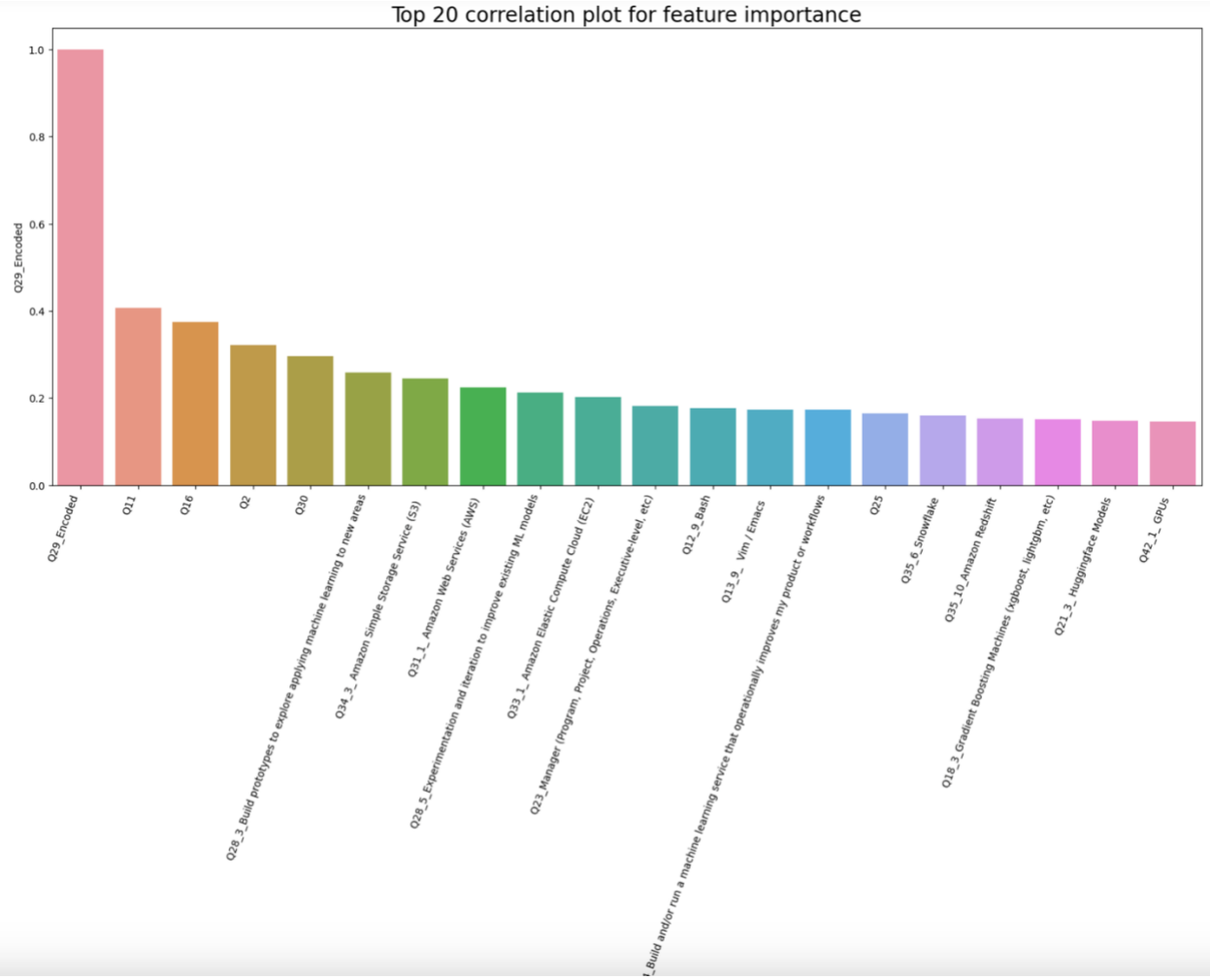
predictions are very less than the actual values. The true target variable has more than 100 observations in each class, but for prediction, the number of predicted values vanished in some classes like 8 and 9.

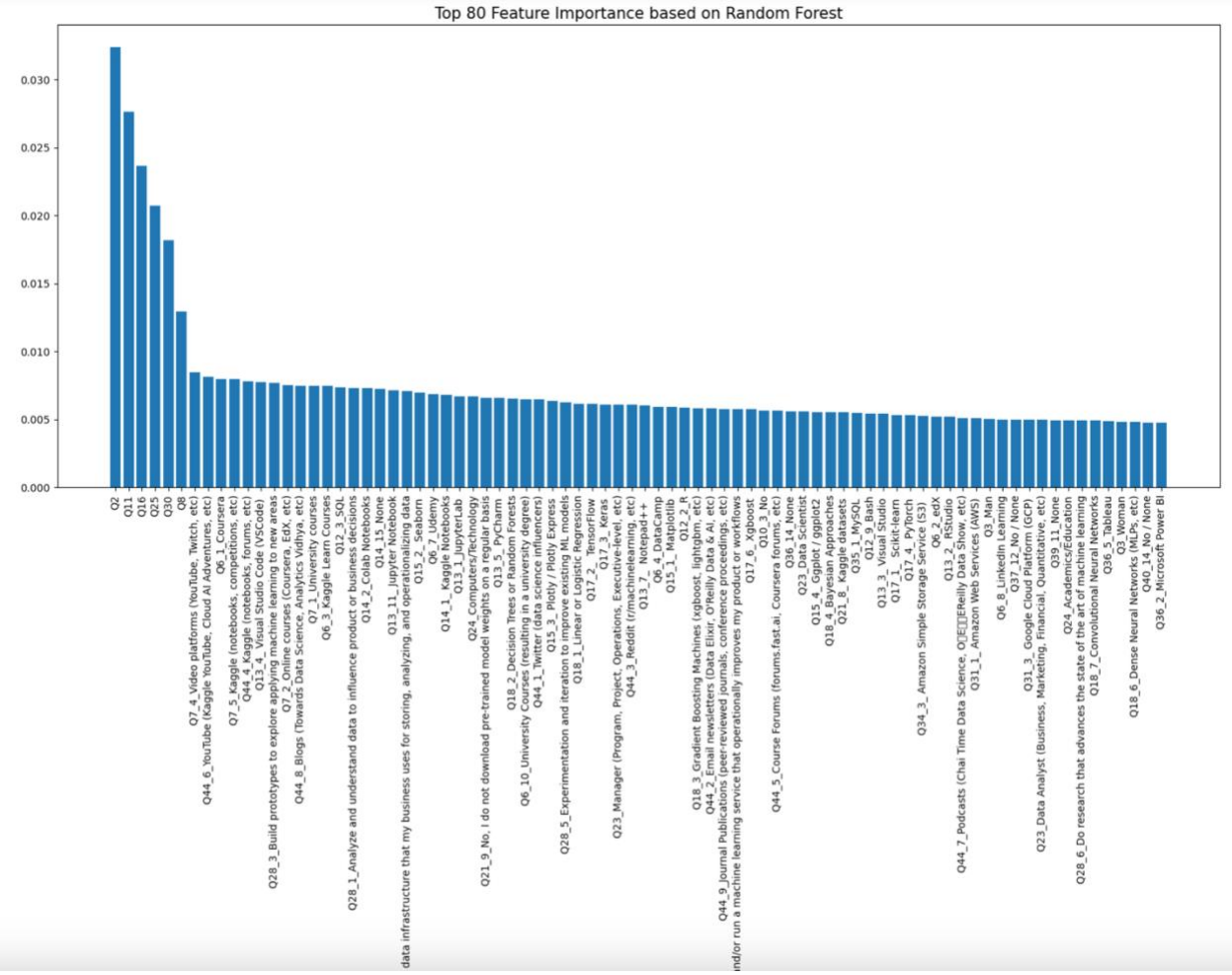
Appendix

Data Cleaning

```
-- Check if all missing values are handled --
Number of missing value within dataframe: 0
```

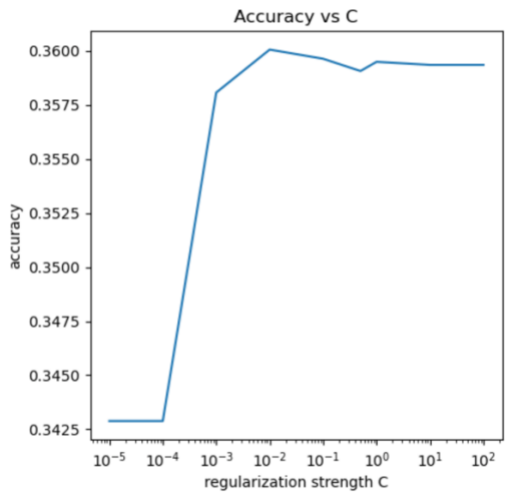
Exploratory data analysis and feature selection

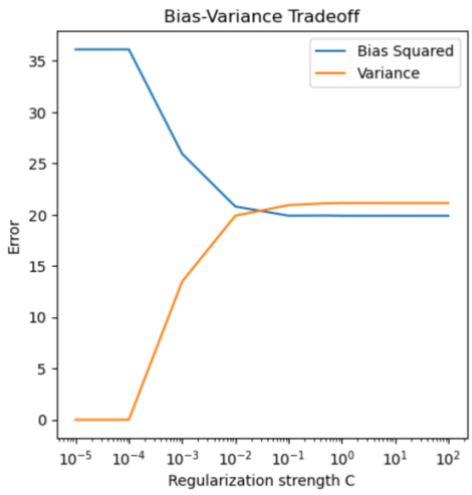




Model Implementation

```
Fold 1: Accuracy: 38.298%
Fold 2: Accuracy: 34.326%
Fold 3: Accuracy: 35.887%
Fold 4: Accuracy: 38.44%
Fold 5: Accuracy: 36.879%
Fold 6: Accuracy: 33.617%
Fold 7: Accuracy: 35.319%
Fold 8: Accuracy: 35.461%
Fold 9: Accuracy: 35.035%
Fold 10: Accuracy: 36.222%
Average Accuracy Score across the folds: 35.948%,
 with a standard deviation of 1.49%
```

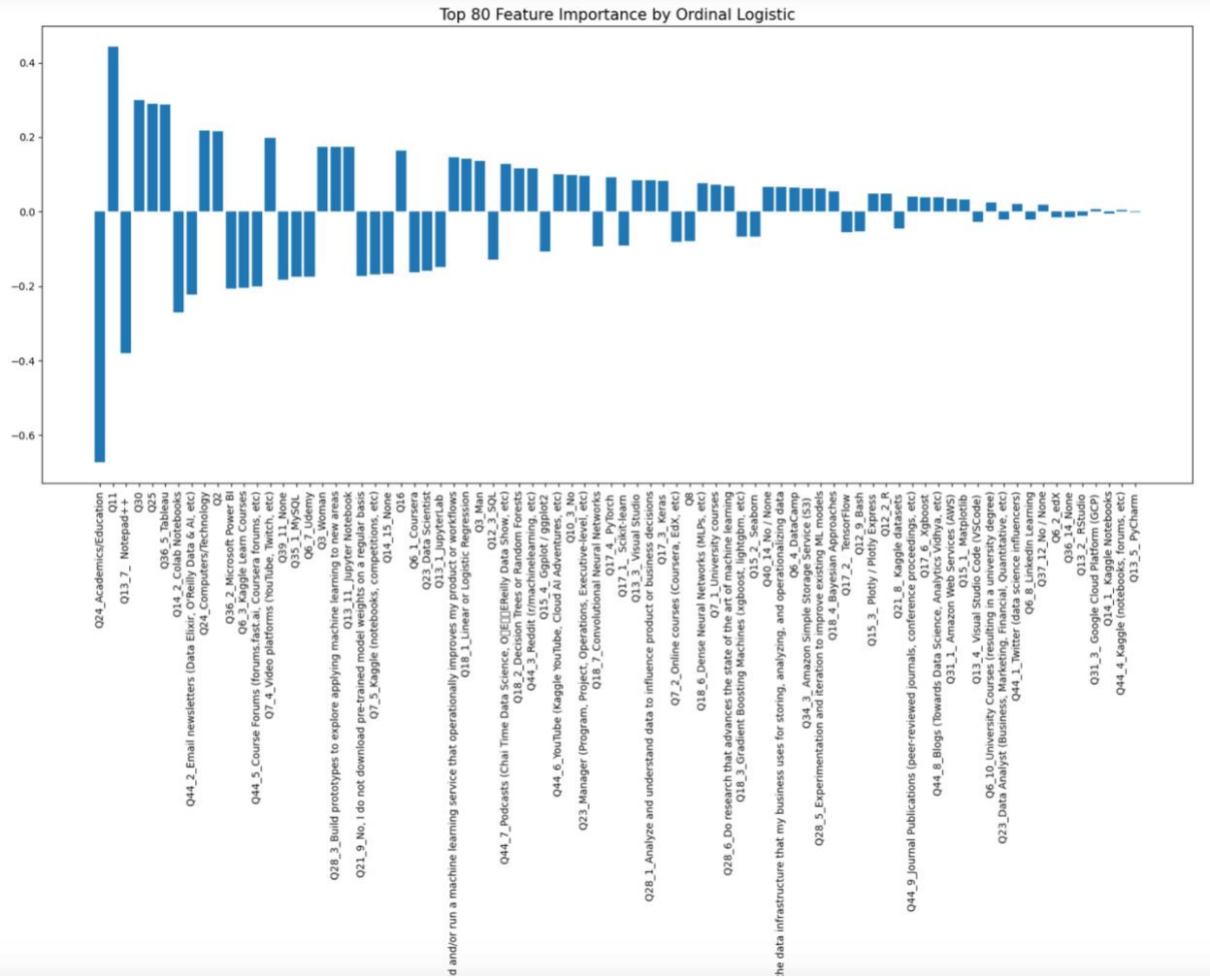




Model tunning

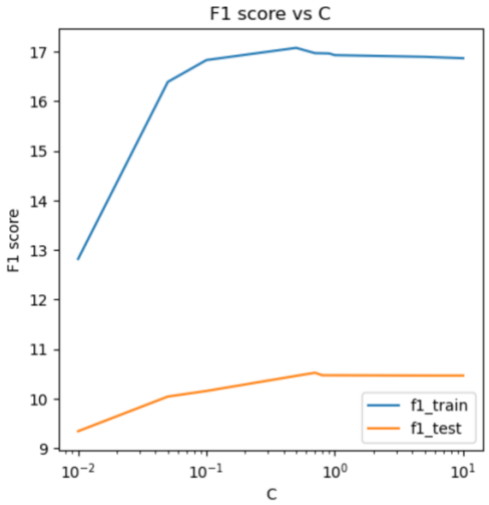
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0.342886	0.097177	0.075897	0.05859	0.055043	0.05405	0.047099	0.045964	0.042276	0.037736	0.036459	0.032061	0.028798	0.026812	0.019152

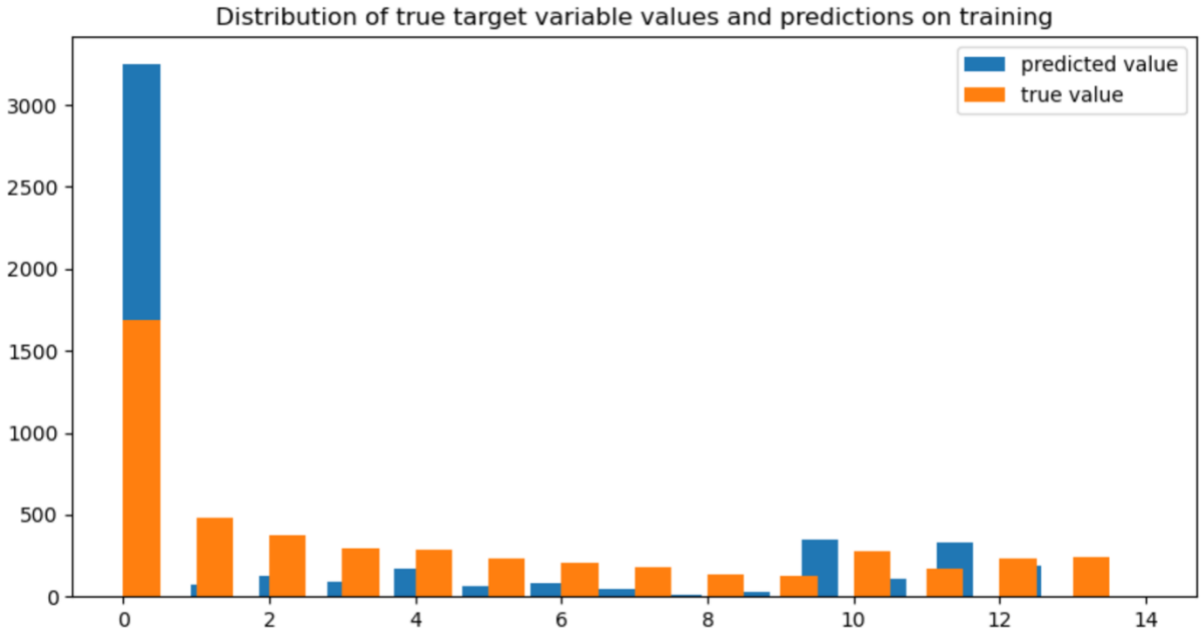
```
{'C': 0.9, 'solver': 'liblinear'}
Best Score: 10.59%(0.882%)

The optimal model uses C=0.9, and a liblinear solver, and has a cross validation score of 10.59% with a standard de
viation of 0.882%
```

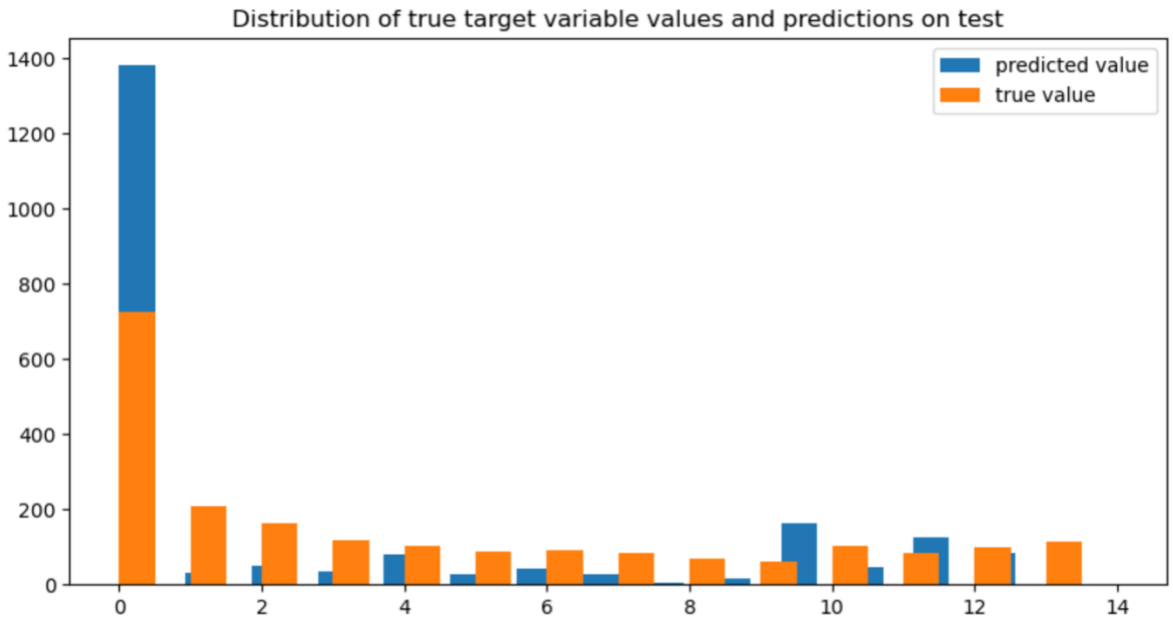


Testing and discussion

```
F1 score of training dataset is: 16.96 %
F1 score of test dataset is: 10.47 %
Accuracy of training dataset is: 39.6 %
Accuracy of test dataset is: 35.37 %
```


F1 score vs C
F1 score
C
f1_train
f1_test


Distribution of true target variable values and predictions on training
predicted value
true value


Distribution of true target variable values and predictions on test
predicted value
true value
1400
1200
1000
800
600
400
200
0
0
2
4
6
8
10
12
14