Team Marvel ML Hackathon Report

- *Aditya Vardhan (IMT2019003)*
- *Archit Sangal (IMT2019012)*

Assumptions:

1. While working on the code which was used in the Hackathon, we forgot to save the local score. So, all we have is the final best local score, and the final best Kaggle score. We were busy with the approach, and we had strict time constraints.

Regression

Preprocessing	Model	Hyperparameters	Local validation score (Mean squared error)	Kaggle score (if submitted)
● Check for duplicated rows (there weren't any) ● Check for missing values (there weren't any) ● Check for datatypes of each column (all were numeric i.e. either int or float) ● Check for skewed data (there weren't any) ● Check for outliers (there weren't any) ● Check for correlation between features (features were mutually independent) ● "ID" was not considered as a feature and thus had no role in model generation. ● Check for categorical features (there weren't any) ● Feature normalization (for all features). ● Scatter plot of each feature vs output (helped us to conclude that a simple linear regression was not enough to generate a good model) ● For model generation and testing the test.csv data was split in a 6:4 ratio for training and testing. ● Checked for various polynomial degrees in a brute force manner. The least error for the train and test was achieved for degree=10.	Simple polynomial regression (NO ridge or lasso)	Polynomial degree=10	Train error = 8.77 Test error = 11.5	Not available

• Inserted a loop for various polynomial degrees and for various values of lambda. • Ran this loop for various train_test_split_data	Ridge polynomial regression	Did the following for "many" train_test_split_data Varied polynomial degree from 2 to 20. Incrementing by 1 in each step. Varied lambda from 0 to 0.02 for each polynomial degree. Incrementing lambda by 0.0001 in each step For least error, we got:- a. Degree = 9 b. Lambda = 0.00048	Train error = 9.26 Test error = 9.48	9.34866
• Inserted a loop for various polynomial degrees and for various values of lambda. • Ran this loop for various train_test_split_data	Lasso polynomial regression	No data available	After a certain number of iterations, I was met with the following warning every single time `ConvergenceWarning: Objective did not converge.`	No data available

Observations:

y-inv almost perfectly fit the degree 2 polynomial regression. However, the mean squared error while considering only y-inv for polynomial degree 2, was far more than what we got when we took both x-inv and y-inv into account, for polynomial degrees greater than 2. This wasn't something that we expected.

Classification

Preprocessing	Model	Hyperparameters	Local validation score	Kaggle score (if submitted)
• Checking for Missing Values (NaN) • Dropping Columns with too many NaN, Threshold which I kept was 20000 for a column • Rows with Nan values are deleted • Checking for '?' marks • Dropped Unique Identification Column and checked for duplicate rows • Split Data between Train and Test (80% vs 20%) as the size of the data was good (more than 2,90,000)	Logistic Regression	solver(liblinear) - we were getting error something of sort maximum number of iterations exceeded while using other solvers. So, we looked it up on documentation here. And it worked. class_weight: Didn't work out great, so we removed it.	No data available but results were not good	No data available
• We noticed that most of the data was Categorical, except 'CARTYPE_13' so we encoded it except the one special column 'CARTYPE_13'. (Didn't work as we forgot to change the data into object data type)	Same as above	Same as above	Same as above	Same as above
• We noticed that 'CARTYPE_13' according to its naming convention should have been a categorical, so we must encode it. (Didn't work as we forgot to change the data into object data type). It was in decimal and had more than 50000 unique values, so we divided it into 13 categories. For example, data value from 0.0 to 3.0 (say) will be put under the category 0. • There was an abnormality in the test data. There were many NaN values, so I repeated the necessary process mentioned above again. • Instead of dropping the data, we used a method 'ffill'. As we can drop columns but not the rows.	Same as above	Same as above	Same as above	Same as above
• According to our estimation, the number of columns would have been more than 500 (roughly) but our columns were 50, so we figured out that one-hot encoding is not taking place. So, we changed the data into object data type. • We noticed that the unique values in columns of	Same as above	Same as above	Got better results than previous results	Got better results than previous results

the test data are not the same as the columns in the train data. Therefore, we removed those columns just so to get a score.				
Same as above	Naive Bayes Bernoulli	Not applicable	Same as above	Same as above
● As results were worse than Naive Bayes Bernoulli, so we switched back to Logistic Regression. ● We noticed that there was a lot of data that was giving zeros as the output. We also noticed that when the answers should have been 1, our model was returning 0 as we removed 50,000 data in order to get unbiased results.	Same as 2 cells above	Same as 2 cells above	Same as 2 cells above	Same as 2 cells above

● We shuffled the data properly, as all the 1's were at the end. We removed a lot of data (nearly 235000 rows that was giving 0 as output), so that we can get an unbiased data for training the data but to test the data we used all the data.
● Deleted some rows that were creating problem in one-hot encoding due to difference in the unique values.
● 'CARTYPE_13' in one-hot encoding was not giving good results, so we removed the conversion of decimal data types to object data type, and we didn't do any one-hot encoding here.
	Same as above	Same as above	`training score = 0.64793537241212`
			
			`test score = 0.64543855133724`
				Public LB: 0.62907
				
				Private LB: 0.63000

LB: Leaderboard

Observations:

1. We had no data description, so we assumed that it was clear that this problem was hit and try. Trying as many permutations and Combinations as possible. If we were not under time constraint, we would want to test some more permutations and combinations.
2. 'CARTYPE_13' was tricky to deal with, it was intuitively a categorical type of feature but was a decimal value with over 50000 unique values.
3. Data has a lot of rows that had 0 as output class which lead to biases, so we needed to take into account the data which is comparatively less biased. We tried this approach; we got some better results. It was told to us that - "Datasets may be imbalanced in nature. You are allowed to use packages to deal with that." But again, we didn't have time, so we took a smaller set of data which gave us the best results of what we had.
4. We did try to find clusters and thought to use different models for each cluster, but the data weren't continuous and we also had a time constraint. So, we weren't able to fully implement this idea.