# Mobile Credit Seller Prediction Model

## Grup C Project

* Aulia Lazuardi Muhammad
* Damario Abdalla
* Muhammad Thowwafi Alfiansyah

Github: https://github.com/thowwafi/pulse-sales-prediction

## Background and problem to be solved (DP1)

Predicting the number of mobile credit seller in an area remains a problem for telco companies. Since each area proposed different opportunities and challenges for a mobile credit seller to thrive on the area. As a telco company, the problem arises when trying to support the mobile credit seller in terms of sales and customer service. If the proportion between the support and the seller are hugely skewed in favour of support, it will lead to inefficiency in terms of costs and time. By recognizing and predicting the supposed available mobile credit seller in that area, the effort and support spent on those area also will be adjusted and prepared accordingly.

## Project Architecture (DP2)

The group intend to predict the mobile credit seller using linear regression as baseline model and other appropriate models, which use features like the area’s capital income, human development index, mass population, etc.

Some of the missing data will be pre-processed and imputed with also external data (BPS data). Then, the data will be normalized before splitting into train & test set. After training the baseline model with the train/test set, Hyperparameter tuning will be commenced to streamline the model. The diagram for the project as below:

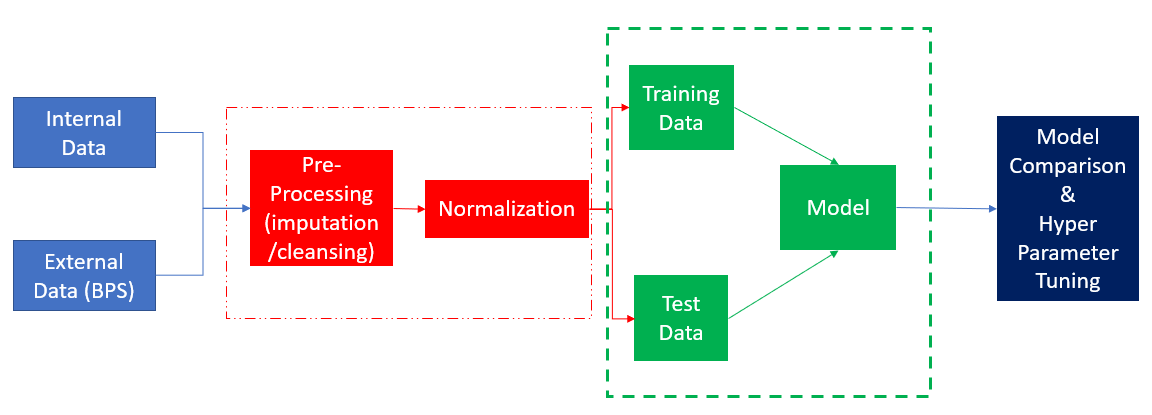


Figure 1. Project Architecture Flow

## Expected Output of Project (DP3)

After Hyper Parameter Tuning, the model would be able to predict the number of the mobile-credit seller of an area with selected features with R-Squared of at least 0.70 and RMSE of 300

## Expected Input Payload (HU1)

The expected input payload would not be continuous data stream, but data batch of the features the user would use.

## Project Output Message Format (HU2)

The output message will be the number of the mobile credit seller predicted of an area.

## Data Insight Gathering (EA1)

At a glance, the dataset has 20 columns which consists 19 features and 1 target. 16 (including the target) of them are numerical while 4 are categorical data. Each column data types could be seen below.

**Numerical**

|  |  |
| --- | --- |
| ID | Pengeluaran\_Riil\_per\_Kapita\_per\_Tahun |
| Jumlah\_Kelurahan\_Desa | Nilai\_UMR |
| PDRB | Jumlah\_Penduduk\_Miskin |
| PDRB\_Per\_Kapita | Jumlah\_Penduduk\_Bekerja |
| Indeks\_Pembangunan\_Manusia | Pengguna\_Internet |
| Jumlah\_Penduduk | Pemilik\_Ponsel |
| Luas\_Wilayah | Pengguna\_Ponsel |
| Dana\_Alokasi\_Umum | Jumlah\_Agen\_Pulsa (Target) |

**Categorical/Object**

|  |
| --- |
| Column |
| Kota\_Kabupaten |
| Provinsi |
| Area |
| Regional |

Firstly, we check the missing values from each column (shown below):

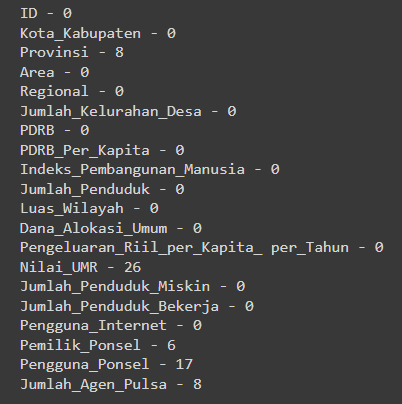


Figure 2. Missing Values Features

So far, only 5 columns out of 20 have some missing values. In which we will dissect and impute the missing values of each feature.

For the “Provinsi” feature, we could use other feature to see the actual value of the missing values. In this case, we could use Kota\_Kabupaten & Regional for imputing the Provinsi values.



Figure 3. Missing Values of Provinsi

From above, anyone could use Google to see which Province belongs to each city (or any geographic-literate Indonesian will do). Imputing the values would pose no problem as seen in the notebook.

As for the Pemilik\_Ponsel features, apparently, all of it came from the same region, “Sulawesi Barat”. In which to impute properly, we will use the Pemilik\_Ponsel median of the Sulawesi Island, since it will be much representative than using all of Indonesian Pemilik\_Ponsel numbers.

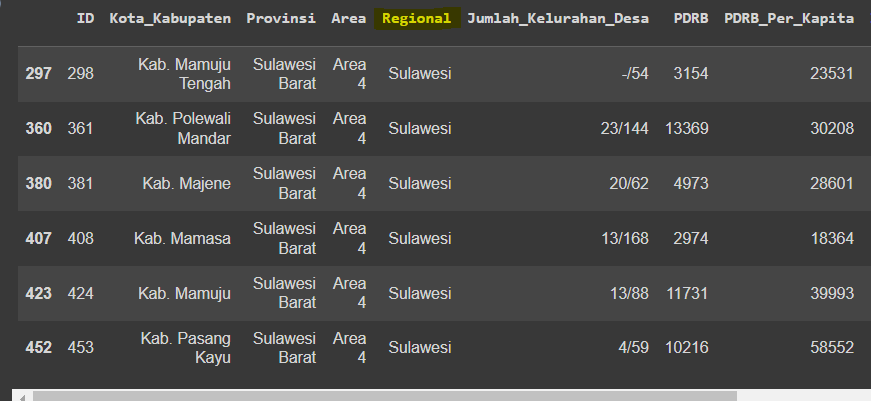


Figure 4. Missing Values of Pemilik\_Ponsel

Fortunately, for the missing values of Pengguna\_Ponsel, the whole bit was coming from Sulawesi Tenggara Province, which we will be using the same trick to use the Sulawesi’s Median of Pengguna\_Ponsel to impute the missing values.



Figure 5. Missing Values of Pengguna Ponsel

As for the missing Nilai\_UMR values, in which 26 of them are missing. In this case, since imputing through median/mean of similar region would not be advisable, we will use external data of the UMR (minimum wage) of each missing province. The provinces are:

* Sumatra Utara
* Sulawesi Utara
* Sulawesi Tengah

Each of the province’s missing UMR\_values will be imputed by googling each of the province. However, even though the missing values were filled. Some of the UMR\_values have 0 value, in which such scenario would be deemed impossible. Hence, we will use the same trick as previously: googling each province’s UMR. The provinces are:

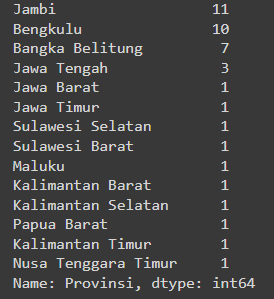


Figure 6. Null values of UMR

After we replace the 0 values in UMR\_values, we realized that we have to check other features that has 0 values but impossible to have to in real life. The features are:

* Dana\_Alokasi\_Umum : 6 zero values
* Jumlah\_Penduduk\_Bekerja: 10 zero values
* Nilai\_UMR: 41 zero values (before imputed)

Similar to previous trick, we will try to impute the zero values with its median with similarities in other features.

For Dana\_Alokasi\_Umum, only Jakarta that has all the zero values. In this case, we will use the median of the java island to replace the zero values. The java island is represented with “Area” feature of Area 2 & Area 3. We will calculate the median of each Area and compute the median of the two numbers.



Figure 7. Null values for Dana\_Alokasi\_Umum

Moving on to Jumlah\_Penduduk\_Bekerja, we will use the Google to find the actual value of the zero values’ province (in this case: Bengkulu, of 1,002,160).

After we impute all the missing and zero values of the features, we will move on to Outlier Checking.

## Pre-processing and Feature Engineering/Selection (EA2)

### Outlier Checking

For all numeric features, we will see the distribution of each column using the boxplot.

|  |
| --- |
| Figure 8. Numerical Features BoxPlot |

It could be seen that almost every feature has a lot of outliers. One of the steps to handle outliers is to remove the outliers, although looking at the data itself, it is unlikely to remove the outlier since the data contains all of Indonesia’s Province. Removing those provinces will be much unwise since the data is very limited.

Similarly, replacing the outliers with median/mean imputation would not be precise as each province/are has its own specific characteristics and data. Hence, we believe to handle the features are to transform them into log-features.

This will become more apparent when we see one of the features’ distributions more closely (PDRB).

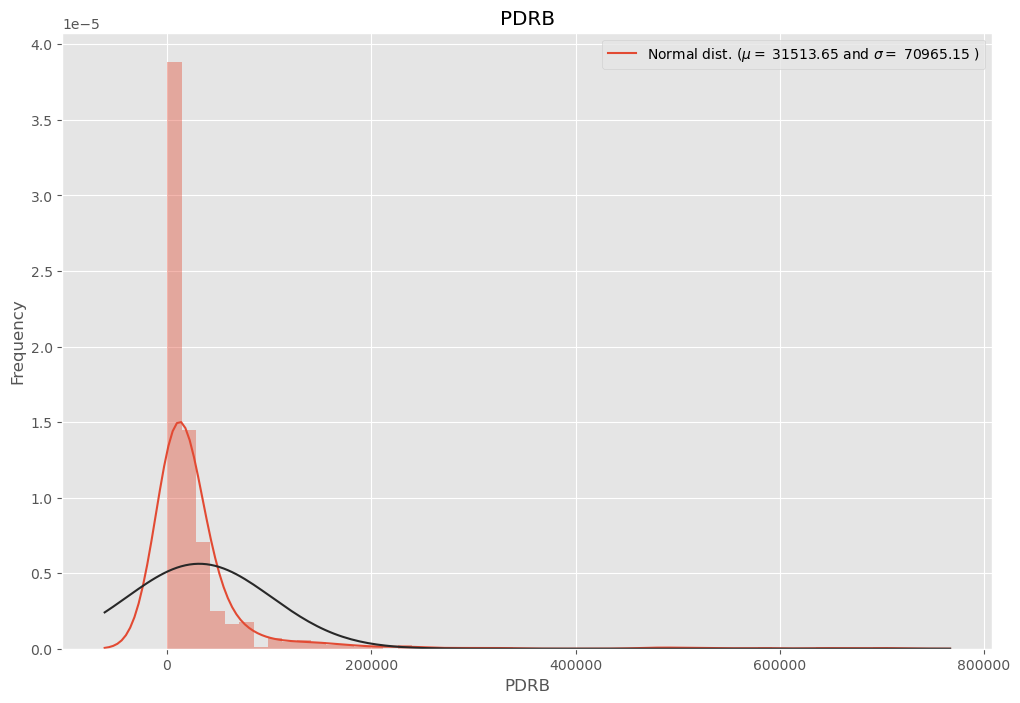


Figure 9. PDRB Distribution

Then, we will convert the values into Log-features with Log(1+p), this will prevent the log transformation to err when log (0) happens. The result of the log transformation could be seen below:

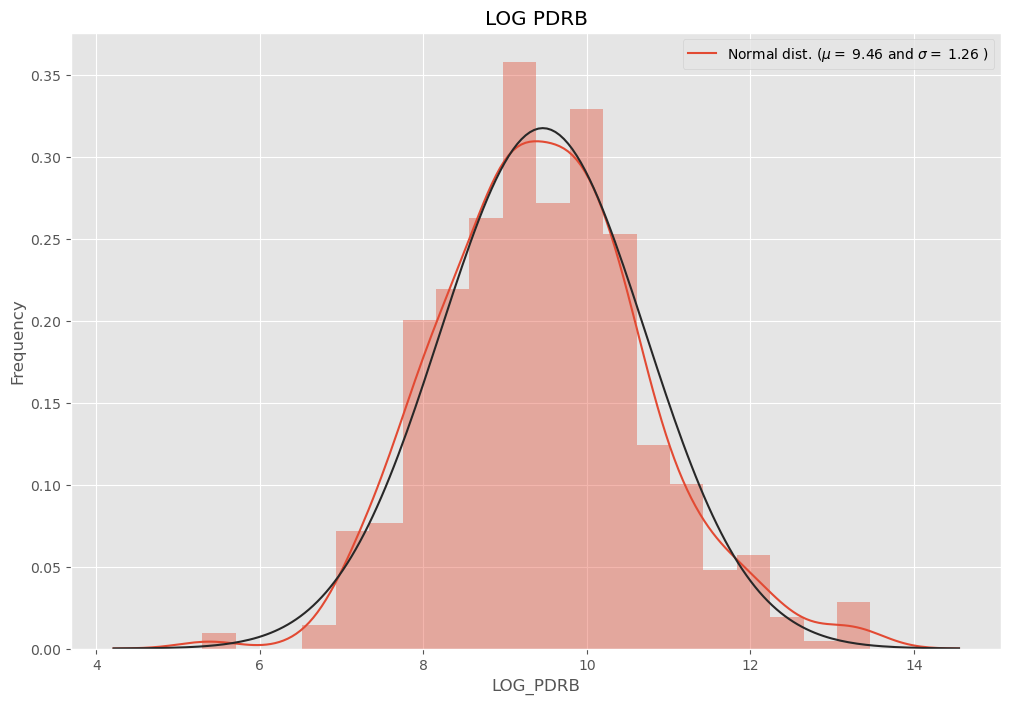


Figure 10. Log-PDRB Distribution

As we can see, the distribution is far more acceptable and the outlier problem seems to vanish. We will proceed to use log-feature for all of our numerical columns.

|  |
| --- |
| Figure 11. Log-Features BoxPlot |

After transformed into log-features, although not all outliers are removed, the distributions are more symmetrical and even.

### Label Encoding for Categorical Value

For categorical features (Regional, Provinsi and Area), we encode it into numerical values. Since the Provinsi and Regional is highly similar, we believed Provinsi feature will be represented by the Regional feature and thus will be removed.

The values to be encoded are:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Area** | **Encode** | | Area 1 | 1 | | Area 2 | 2 | | Area 3 | 3 | | Area 4 | 4 | | Area 5 | 5 | | |  |  | | --- | --- | | **Regional** | **Encode** | | Sumbagsel | 1 | | Lampung | 2 | | Sumbagut | 3 | | Sumbagteng | 4 | | Jabar | 5 | | Jabo Inner | 6 | | Jabo Outer | 7 | | Jatim | 8 | | Jateng | 9 | | Sulawesi | 10 | | Kalimantan | 11 | | Malpua | 12 | | Balnus | 13 | |

### Features Correlation Heat Map

After we transformed the features, we could see the correlation heat map of each of the features to see if we could remove some of the features before feeding it to the model.

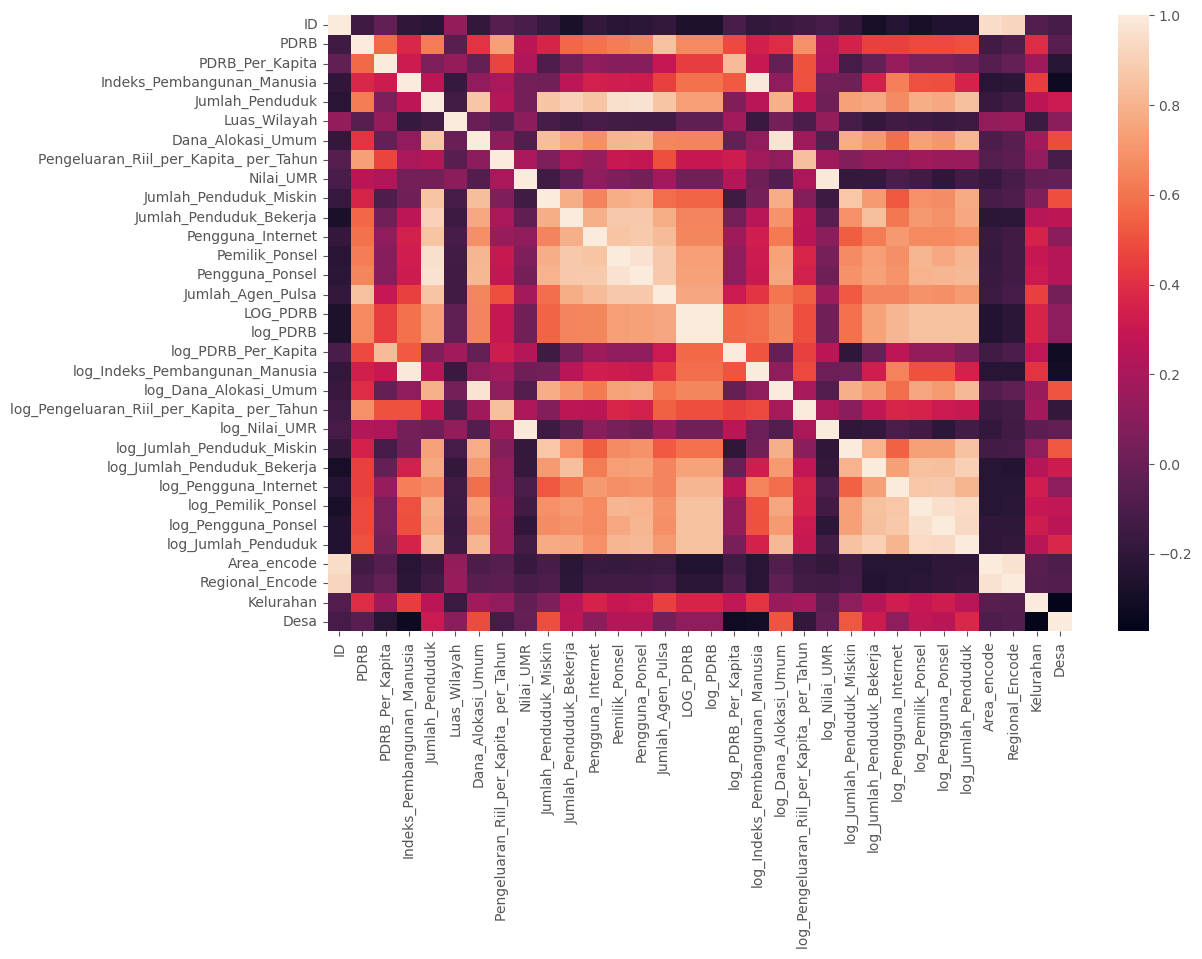


Figure 12. Features Correlation Heat Map

As we could see, some of the features are highly correlated with each other as expected. However, we will focus on features that highly correlated with the target (Jumlah\_Agen\_Pulsa). In this case the correlation between the features to the target does not vary much.

Hence, we will try to model all the log-features and then try to select the features by forward-feeding and backward feeding before train other models.

### Modelling

For the baseline model, we will use the skicit-learn library’s LinearRegression and try to see the outstanding features with Lasso. All the features used are the log-features and the target is the Jumlah\_Agen\_Pulsa. The ratio between the training set & test set is 1:3 with random\_state= 42.

After training the first linear regression model, we got the results of:

|  |  |
| --- | --- |
| R2 Score | 0.6177 |
| RMSE (Root Mean Square Error) | 315.37 |

Even though the model itself is not as bad for a baseline model, it is still fall short to achieve our goal (0.7). However, we could look at the coefficient of the baseline model, which yields:

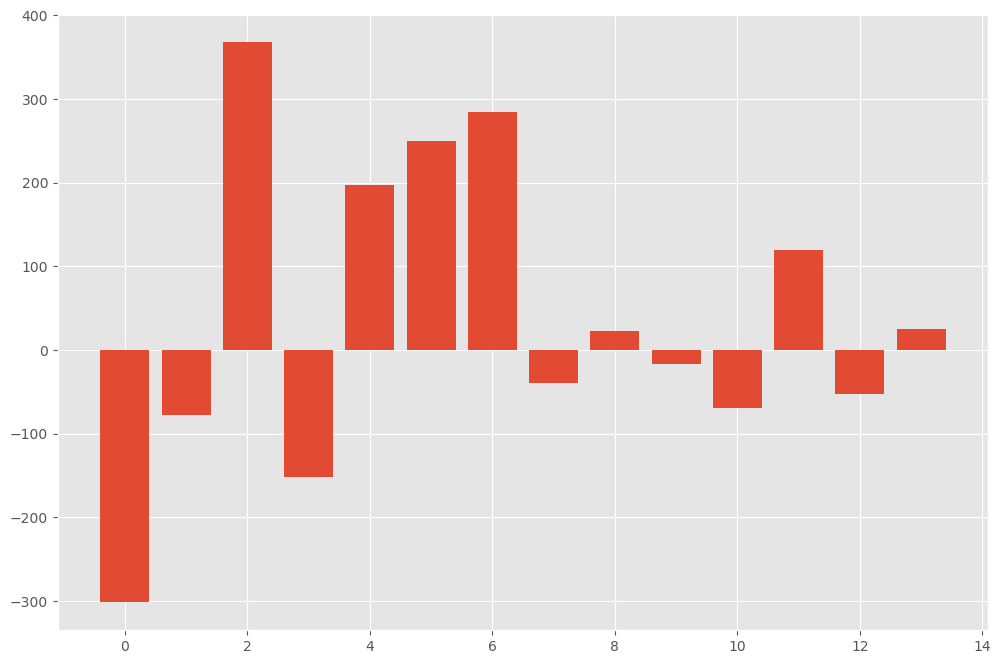


Figure 13. Linear Regression Coefficient rank

As for the actual features name and score:

|  |  |  |
| --- | --- | --- |
| **Feature Names** | **Feature Label** | **Score** |
| Log\_PDRB\_Per\_Kapita | Feature 0 | -301.741 |
| Log\_indeks\_Pembangunan\_Manusia | Feature 1 | -78.028 |
| Log\_PDRB | Feature 2 | 367.723 |
| Log\_jumlah\_Penduduk | Feature 3 | -151.655 |
| Log\_Dana\_Alokasi\_Umum | Feature 4 | 197.684 |
| Log\_Pengeluaran\_Riil\_per\_Kapita\_per\_tahun | Feature 5 | 249.91 |
| Log\_nilai\_UMR | Feature 6 | 283.989 |
| Log\_jumlah\_Penduduk\_Miskin | Feature 7 | -39.893 |
| Log\_jumlah\_Penduduk\_Bekerja | Feature 8 | 22.394 |
| Log\_Pengguna\_Internet | Feature 9 | -16.741 |
| Log\_Pemilik\_Ponsel | Feature 10 | -69.362 |
| Log\_pengguna\_Ponsel | Feature 11 | 119.615 |
| Area\_encode | Feature 12 | -52.610 |
| Regional\_encode | Feature 13 | 24.414 |

At a Glance, we could tell that the most important feature will be Feature 2 (Log\_PDRB), followed by Feature 0 (Log\_PDRB\_per\_Kapita) and then Feature 6 (Log\_Nilai\_UMR). The link between feature 2 and feature 0 is very obvious since both of them includes PDRB ( Produk Domestik Regional Bruto) and the Region’s Minimum Wage also made sense to predict the number of the region’s mobile-credit seller.

Least interesting feature consists of Feature 8, 9 & 13 (Number of working forces, Internet User Number and Regional Encode). Although the first 2 feature are quite a surprise to see it lacks power to predict the target especially Feature 8, the last feature made sense since the regional encode would not matter much since other features will play more important role. This will hold the same with Feature 12 (Area Encode). As for the Internet User number we believe since nowadays people who uses internet does not require much for mobile credit, hence the minimum correlation.

Next, we will try lasso regression to see if our initial hypothesis of feature importance will hold. For the R score, the value still holds the same at 0.617, as for the Feature Importance:

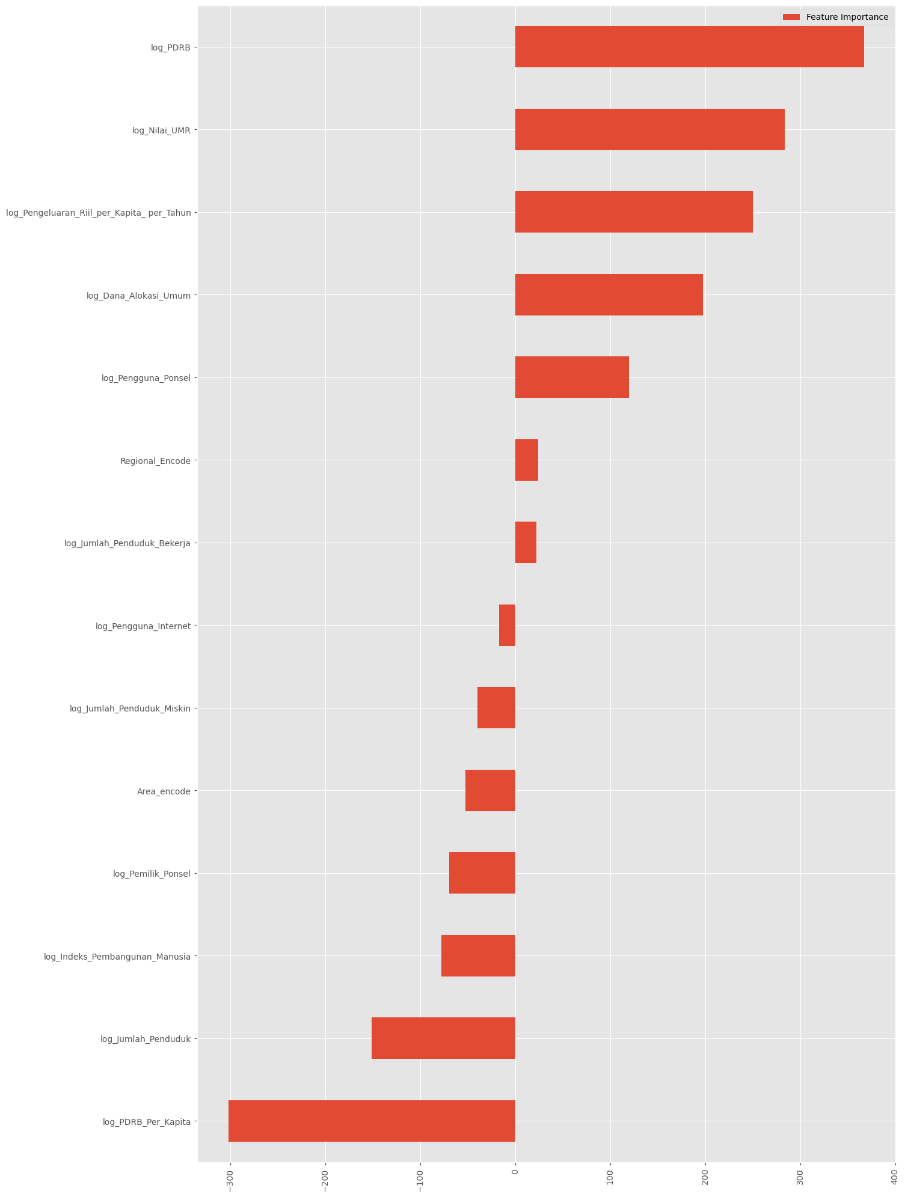


Figure 14. Lasso Linear Regression Feature Rank

And for the feature rank and value:

|  |  |  |  |
| --- | --- | --- | --- |
| Log\_PDRB | 367.486 | Log\_Pengguna\_Internet | -16.746 |
| Log\_nilai\_UMR | 283.966 | Log\_jumlah\_Penduduk\_Miskin | -39.87 |
| Log\_Pengeluaran\_Riil\_per\_Kapita\_per\_tahun | 249.902 | Area\_encode | -52.594 |
| Log\_Dana\_Alokasi\_Umum | 197.644 | Log\_Pemilik\_Ponsel | -69.350 |
| Log\_pengguna\_Ponsel | 119.565 | Log\_indeks\_Pembangunan\_Manusia | -77.589 |
| Regional\_encode | 24.409 | Log\_jumlah\_Penduduk | -151.391 |
| Log\_jumlah\_Penduduk\_Bekerja | 22.392 | Log\_PDRB\_Per\_Kapita | -301.508 |

As we could see, the Lasso regression feature importance hold the same rank as the basic linear regression. Hence, it will be wise to use the important feature listed. However, we will try another model to see if we could perform better.

Next, we will use Decision Tree Regressor from the scikit-learn library. The first Tree Regressor has parameters listed below:

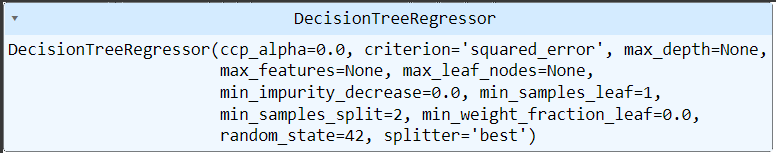


Figure 15. Decision Tree parameters

Which yields result:

|  |  |
| --- | --- |
| R2 Score | 0.742 |
| RMSE (Root Mean Square Error) | 258.82 |

The result for the Tree Regressor yields much better results for the R2 and the RMSE of the target are lower than the baseline linear regression model. This also fulfilled our objective score of this project. For improving the model, we also tried to use Hyper parameter Tuning with GridSearchCV to find the best combination of each feature. However, we did not include the result into this report due to the time limitation.

## Conclusion and Notes for Further Improvement (CR1)

To conclude, the best model to predict the number of mobile-credit seller in a region/area is to use the regression tree with log-features. The most important features would be the PDRB, PDRB/Kapita & UMR, which made a lot of sense regarding the region’s economic purchasing power and its correlation with the number of mobile-credit seller

For feature engineering, some new features such as ratio between Jumlah\_Penduduk\_Bekerja & Jumlah\_Penduduk also can be explored too. As for normalizing all the features, we believe it only have limited benefits but further investigation could prove us wrong.

Other suggestions point out that instead using label encoding for categorical features like Area & Regional, one-hot encoding also could be applied. In our case, since we are trying to avoid too many features to work with, this idea could be implemented for further improvement.

As for the Hyper Parameter Tuning, other not-so-costly methods such as randomsearchCV could be implemented instead of the exhaustive method like GridSearchCV.

## List of Reference Links (CR2)

* [**https://towardsdatascience.com/hyperparameter-tuning-in-python-21a76794a1f7**](https://towardsdatascience.com/hyperparameter-tuning-in-python-21a76794a1f7)
* **https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/**
* [**http://rasbt.github.io/mlxtend/user\_guide/feature\_selection/SequentialFeatureSelector/#example-6-feature-selection-with-fixed-trainvalidation-splits**](http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/#example-6-feature-selection-with-fixed-trainvalidation-splits)
* [**https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html**](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html)
* **Github Project: https://github.com/thowwafi/pulse-sales-prediction**