ML Assessment 5

Rodrigo Velázquez E.

Leandro Sartini de Campos

Mohammed Maaz

Aman Nain

Prakash Pun

Saraka Sahaswin

Bibek Ranjit

AI & DS, Loyalist College of Toronto

M07 Group 1: AISC1003 - Machine Learning I

Prof. Bhavik Gandhi

Churn Prediction

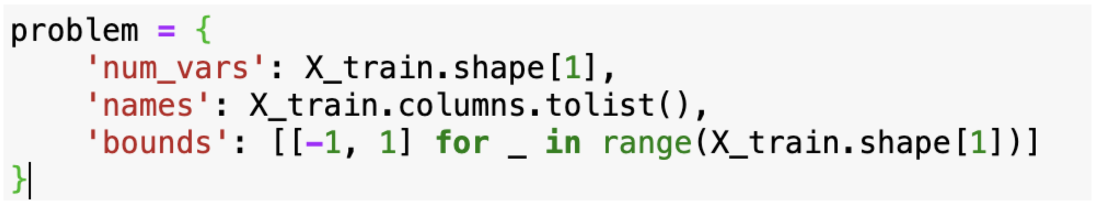
Contents

Recap from Week 11 Assignment

1. EDA
2. Correlation
3. One-Hot Encoding
4. Scaling
5. Balance Data
6. logistic regression and SVM model
7. Built a confusion matrix and AUC plot for the model

Sensitivity Analysis on Linear Regression

1. We define a problem specification for sensitivity analysis, specifying the number of variables, their names, and their bounds.



```
problem = {
    'num_vars': X_train.shape[1],
    'names': X_train.columns.tolist(),
    'bounds': [[-1, 1] for _ in range(X_train.shape[1])]
}
```

   - 'num_vars': specifies the number of variables in the sensitivity analysis problem.
   - 'names': List of variable or column names
   - 'bonds': Specifies the bounds (ranges) for each variable in the sensitivity analysis.
2. Generate input samples using the Sobol sampling where each sample represents a combination of input parameters (values for each variable) that will be used to evaluate the model for sensitivity analysis.
3. Evaluate our logistic regression model for each set of input parameters.
4. Performed Sensitivity analysis using SALib's 'sobol.analyze()' function.

```
# Placeholder for model predictions
Y = np.zeros(param_values.shape[0])

for i, X in enumerate(param_values):
    # Rescale or transform X if necessary to match your model's input
    # For demonstration, assuming direct use:
    Y[i] = logr.predict_proba(X.reshape(1, -1))[:,1]
```

```
Si = sobol.analyze(problem, Y)

# Si is a dictionary containing keys for different metrics, such as 'S
print(Si['S1'])
```

```
[ 1.34180506e-02 -2.48957323e-04  5.73741172e-03  8.18103333e-02
  2.49319014e-02  4.20366291e-04  1.45721941e-03  7.67422522e-05
 -7.43184602e-05  6.21061839e-01  1.44215702e-05  1.96763819e-04
  2.17716619e-04  5.27203498e-03  5.00442403e-03  1.48973346e-04
  7.29393557e-04  7.09524937e-04  2.00245191e-03  3.67994616e-04
  1.03917798e-05 -6.08924060e-05  6.39555478e-04 -5.25461531e-04
  1.56368489e-04  1.14219859e-04  2.62423961e-04]
```

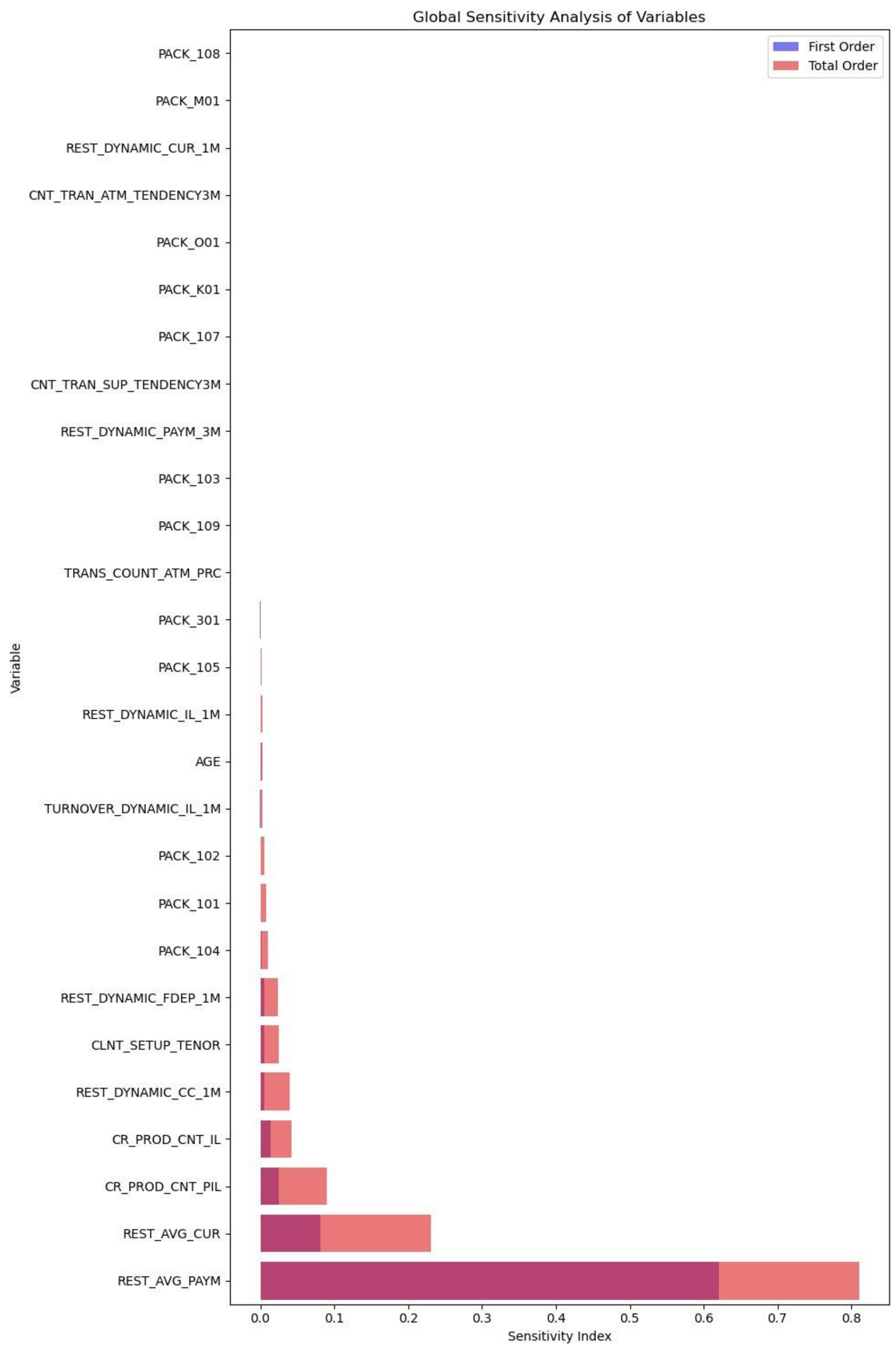
- ‘S1’: represents the first-order sensitivity indices, which tells the individual effects of input variables on the model output.
- ‘ST’: represents the total-order sensitivity indices, which measure the total effect of the input variable on the model output.
- ‘S2’: represents the second-order sensitivity indices, which tells the pairwise interaction between two input variables.
- ‘sensitivity_conf’: contains the confidence intervals for the sensitivity indices computed.

5. We then create a ‘df_sensitivity_sorted’ variable with input variables ordered based on the total-order sensitivity, from least to most influential. And then plot the visual representation of the sensitivity analysis result, allowing for easy comparison of the first-order and total-

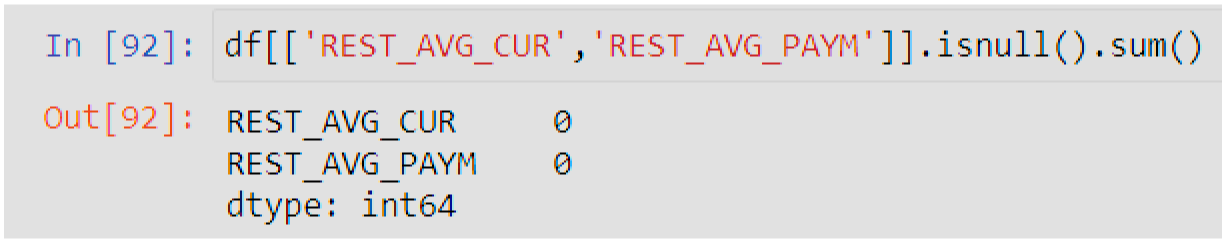
order sensitivity indices for each input variable.

```
df_sensitivity = pd.DataFrame({
    'Variable': problem['names'],
    'First Order Sensitivity': Si['S1'],
    'Total Order Sensitivity': Si['ST']
})

# Sorting the DataFrame by Total Order Sensitivity for better visualization
df_sensitivity_sorted = df_sensitivity.sort_values(by='Total Order Sensitivity', ascending=True)
```

```
plt.figure(figsize=(10, 15))  # Adjust the figure size as necessary

# First Order Sensitivity
sns.barplot(x='First Order Sensitivity', y='Variable', data=df_sensitivity_sorted, label='First Order', color='blue'

# Total Order Sensitivity
sns.barplot(x='Total Order Sensitivity', y='Variable', data=df_sensitivity_sorted, label='Total Order', color='red',

plt.xlabel('Sensitivity Index')
plt.ylabel('Variable')
plt.title('Global Sensitivity Analysis of Variables')
plt.legend()

plt.tight_layout()
plt.show()
```



As we can see most variables have low sensitivity indices, however, “REST_AVG_CUR” and especially “REST_AVG_PSR” show significantly higher sensitivity which means these variables have a greater impact on the model output.



```
In [92]: df[['REST_AVG_CUR','REST_AVG_PAYM']].isnull().sum()

Out[92]: REST_AVG_CUR     0
         REST_AVG_PAYM    0
         dtype: int64
```

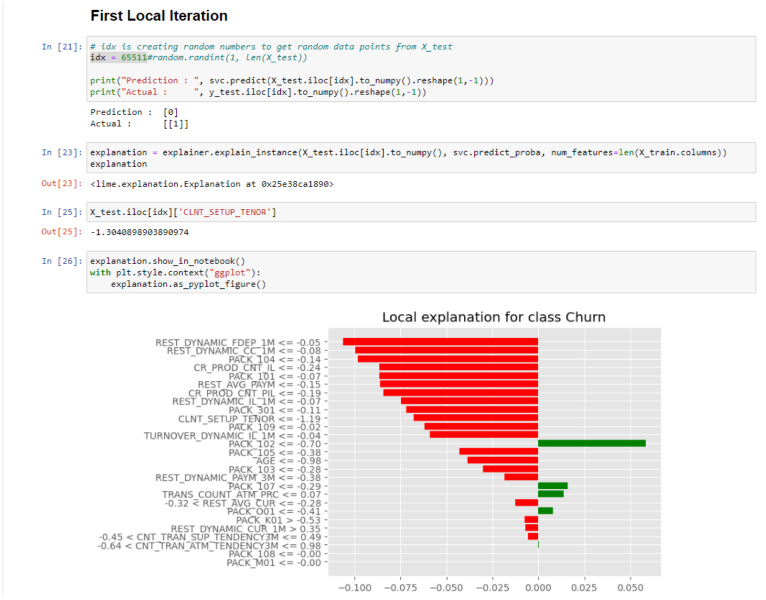
And when we check this on the original Data frame, it’s not a variable that we treated, but it may be a variable that creates other variables.

Interpretation of SVM (LIME and SHAP)

LIME

We are generating values for idx with random.randint to run LIME on those specific instances as LIME can only be used for individual predictions.

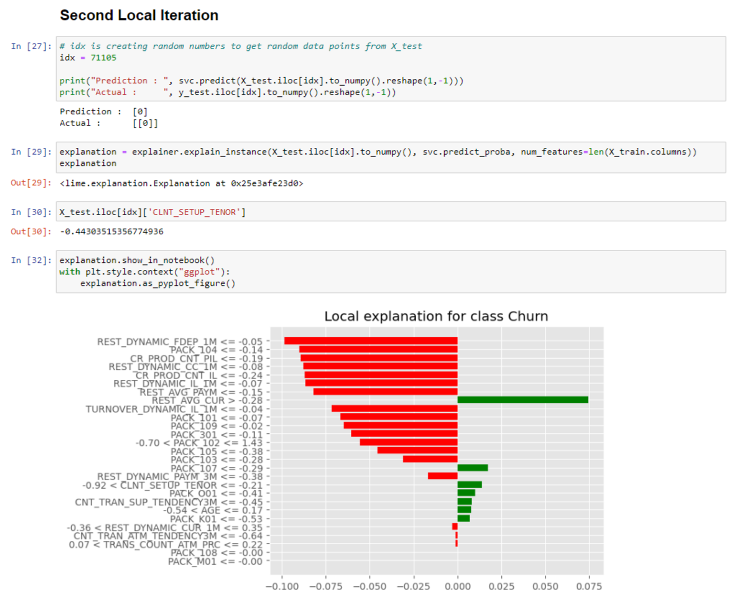
Following are 3 instances generated through LIME.



First Local Iteration

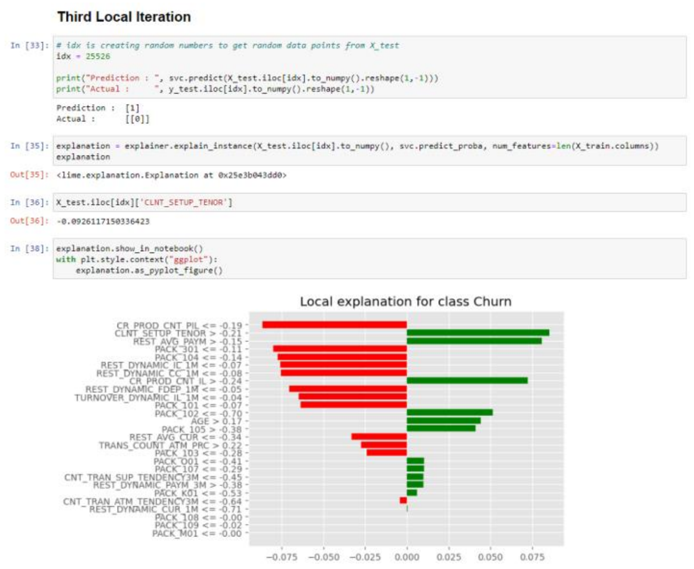
```
In [21]: # idx is creating random numbers to get random data points from X_test
         idx = 65511#random.randint(1, len(X_test))

         print("Prediction : ", svc.predict(X_test.iloc[idx].to_numpy().reshape(1,-1)))
         print("Actual :     ", y_test.iloc[idx].to_numpy().reshape(1,-1))

         Prediction :  [0]
         Actual :      [[1]]

In [23]: explanation = explainer.explain_instance(X_test.iloc[idx].to_numpy(), svc.predict_proba, num_features=len(X_train.columns))
         explanation

Out[23]: <lime.explanation.Explanation at 0x25e38ca1890>

In [25]: X_test.iloc[idx]['CLNT_SETUP_TENOR']

Out[25]: -1.3040898903890974

In [26]: explanation.show_in_notebook()
         with plt.style.context("ggplot"):
             explanation.as_pyplot_figure()
```

Local explanation for class Churn
REST_DYNAMIC_FDEP_1M <= -0.05
REST_DYNAMIC_CC_1M <= -0.08
PACK_104 <= -0.14
CR_PROD_CNT_IL <= -0.24
PACK_101 <= -0.07
REST_AVG_PAYM <= -0.15
CR_PROD_CNT_PIL <= -0.19
REST_DYNAMIC_IL_1M <= -0.07
PACK_301 <= -0.11
CLNT_SETUP_TENOR <= -1.19
PACK_109 <= -0.02
TURNOVER_DYNAMIC_IL_1M <= -0.04
PACK_102 <= -0.70
PACK_105 <= -0.38
AGE <= -0.98
PACK_103 <= -0.28
REST_DYNAMIC_PAYM_3M <= -0.38
PACK_107 <= -0.29
TRANS_COUNT_ATM_PRC <= 0.07
-0.32 < REST_AVG_CUR <= -0.28
PACK_O01 <= -0.41
PACK_K01 > -0.53
REST_DYNAMIC_CUR_1M > 0.35
-0.45 < CNT_TRAN_SUP_TENDENCY3M <= 0.49
-0.64 < CNT_TRAN_ATM_TENDENCY3M <= 0.98
PACK_108 <= -0.00
PACK_M01 <= -0.00
−0.100
−0.075
−0.050
−0.025
0.000
0.025
0.050



Second Local Iteration

```
In [27]: # idx is creating random numbers to get random data points from X_test
         idx = 71105

         print("Prediction : ", svc.predict(X_test.iloc[idx].to_numpy().reshape(1,-1)))
         print("Actual :     ", y_test.iloc[idx].to_numpy().reshape(1,-1))

         Prediction :  [0]
         Actual :      [[0]]

In [29]: explanation = explainer.explain_instance(X_test.iloc[idx].to_numpy(), svc.predict_proba, num_features=len(X_train.columns))
         explanation

Out[29]: <lime.explanation.Explanation at 0x25e3afe23d0>

In [30]: X_test.iloc[idx]['CLNT_SETUP_TENOR']

Out[30]: -0.44303515356774936

In [32]: explanation.show_in_notebook()
         with plt.style.context("ggplot"):
             explanation.as_pyplot_figure()
```



Third Local Iteration

```
In [33]: # idx is creating random numbers to get random data points from X_test
         idx = 25526

         print("Prediction : ", svc.predict(X_test.iloc[idx].to_numpy().reshape(1,-1)))
         print("Actual :     ", y_test.iloc[idx].to_numpy().reshape(1,-1))

         Prediction :  [1]
         Actual :      [[0]]

In [35]: explanation = explainer.explain_instance(X_test.iloc[idx].to_numpy(), svc.predict_proba, num_features=len(X_train.columns))
         explanation

Out[35]: <lime.explanation.Explanation at 0x25e3b043dd0>

In [36]: X_test.iloc[idx]['CLNT_SETUP_TENOR']

Out[36]: -0.0926117150336423

In [38]: explanation.show_in_notebook()
         with plt.style.context("ggplot"):
             explanation.as_pyplot_figure()
```

Local Feature Importance

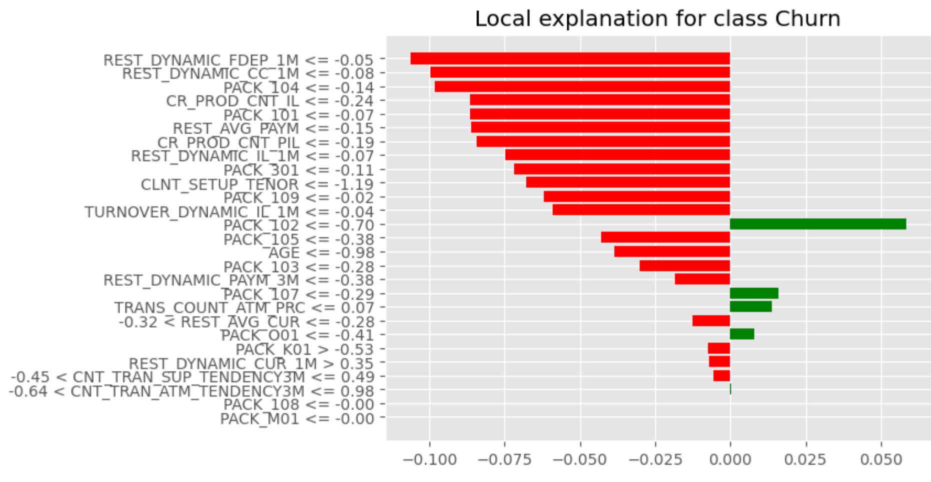
Out of the above three instances, we can see that in ⅔ scenarios, feature PACK_104(One Hot encoded from a given feature) has caused the prediction to move towards 0. While features like PACK_107 contribute to the first two instances positively, the contributions of other features have moved it to the negative side.

In the 3rd instance, as the prediction is positive we can clearly observe features which have contributed positively for the model to predict 1.

Local Instance 1

idx = 65511

Prediction : [0]   Actual : [1]



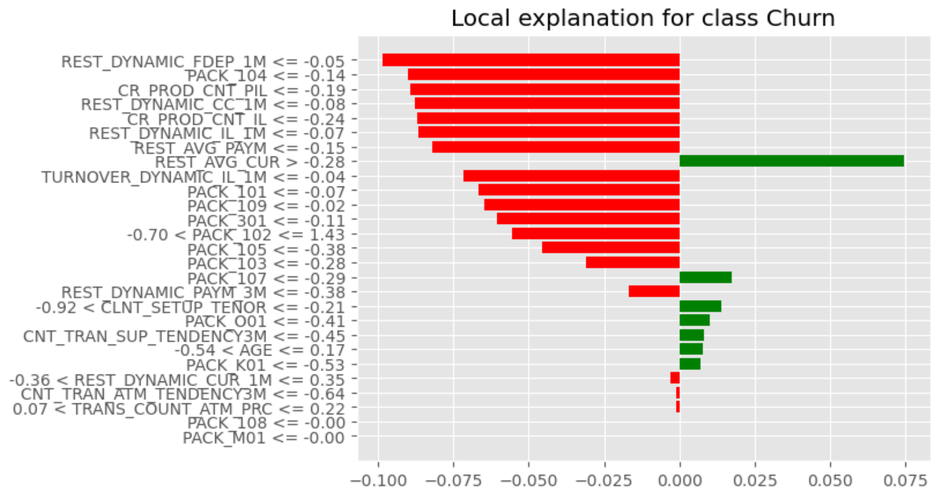
```
CR_PROD_CNT_IL              -0.243936
TURNOVER_DYNAMIC_IL_1M      -0.044953
REST_DYNAMIC_FDEP_1M        -0.050726
REST_AVG_CUR                -0.299455
CR_PROD_CNT_PIL             -0.192076
TRANS_COUNT_ATM_PRC          0.067634
AGE                         -1.063486
REST_DYNAMIC_IL_1M          -0.068469
REST_DYNAMIC_CUR_1M          1.147949
REST_AVG_PAYM               -0.151094
CNT_TRAN_SUP_TENDENCY3M      0.488931
REST_DYNAMIC_PAYM_3M        -0.375851
CNT_TRAN_ATM_TENDENCY3M      0.980249
CLNT_SETUP_TENOR            -1.304090
REST_DYNAMIC_CC_1M          -0.083145
PACK_101                    -0.072293
PACK_102                    -0.700611
PACK_103                    -0.275304
PACK_104                    -0.139191
PACK_105                    -0.380981
PACK_107                    -0.292075
PACK_108                    -0.002740
PACK_109                    -0.015380
PACK_301                    -0.109144
PACK_K01                     1.900770
PACK_M01                    -0.004746
PACK_O01                    -0.406695
```

In this datapoint we can see that, while values like REST_DYNAMIC_CUR_1M,CNT_TRAN_ATM_TENDENCY3M have higher values, they have lesser significance in the model affecting the prediction. While features like PACK,REST_DYNAMIC_PAYM_3M have affected prediction positively, due to the high weightage towards the negative side, the model has predicted negative.

Local Instance 2

idx = 71105

Prediction : [0]   Actual : [0]



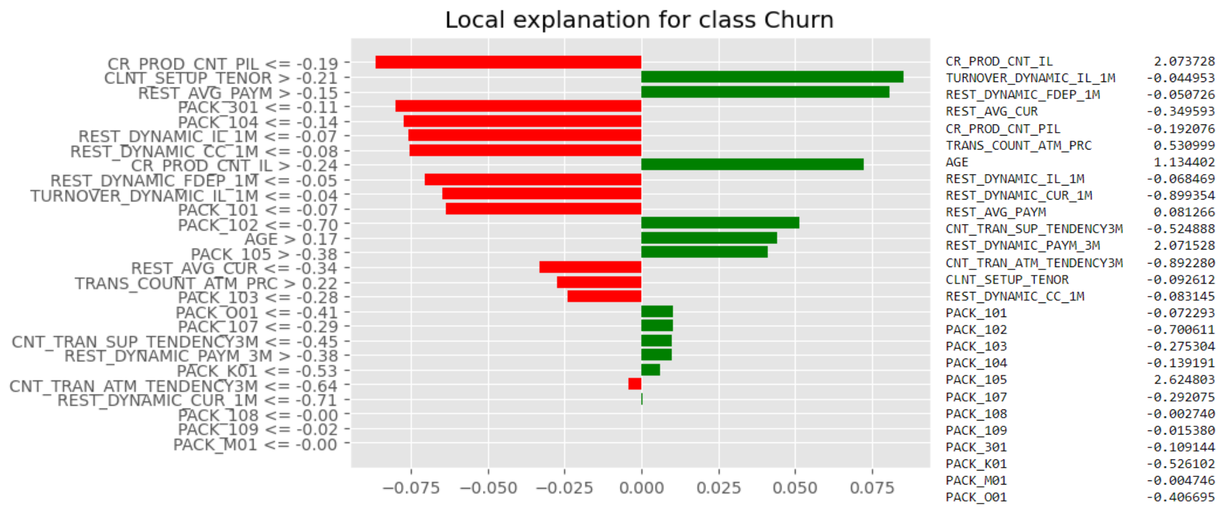
```
CR_PROD_CNT_IL              -0.243936
TURNOVER_DYNAMIC_IL_1M      -0.044953
REST_DYNAMIC_FDEP_1M        -0.050726
REST_AVG_CUR                -0.193918
CR_PROD_CNT_PIL             -0.192076
TRANS_COUNT_ATM_PRC          0.216669
AGE                         -0.096416
REST_DYNAMIC_IL_1M          -0.068469
REST_DYNAMIC_CUR_1M          0.048786
REST_AVG_PAYM               -0.151094
CNT_TRAN_SUP_TENDENCY3M     -0.451908
REST_DYNAMIC_PAYM_3M        -0.375851
CNT_TRAN_ATM_TENDENCY3M     -0.635740
CLNT_SETUP_TENOR            -0.443035
REST_DYNAMIC_CC_1M          -0.083145
PACK_101                    -0.072293
PACK_102                     1.427326
PACK_103                    -0.275304
PACK_104                    -0.139191
PACK_105                    -0.380981
PACK_107                    -0.292075
PACK_108                    -0.002740
PACK_109                    -0.015380
PACK_301                    -0.109144
PACK_K01                    -0.526102
PACK_M01                    -0.004746
PACK_O01                    -0.406695
```

In this instance, the model has predicted correctly as 0. While the value for REST_AVG_CUR is -0.193918, looking at this graph we can infer that, for the model this feature is heavily weighted and positively affected to the target. However as the previous instance, as the negative affected features are higher and have more weight, the model has predicted negative.

Local Instance 3

idx = 25526

Prediction : [1] *Actual : [0]*

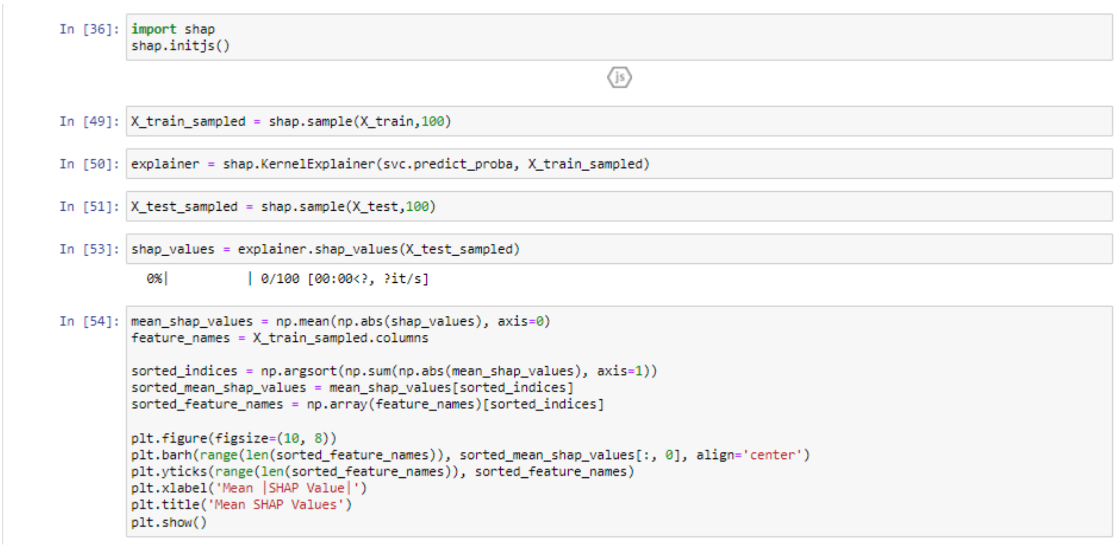


```
CR_PROD_CNT_IL              2.073728
TURNOVER_DYNAMIC_IL_1M     -0.044953
REST_DYNAMIC_FDEP_1M       -0.050726
REST_AVG_CUR               -0.349593
CR_PROD_CNT_PIL            -0.192076
TRANS_COUNT_ATM_PRC         0.530999
AGE                         1.134402
REST_DYNAMIC_IL_1M         -0.068469
REST_DYNAMIC_CUR_1M        -0.899354
REST_AVG_PAYM               0.081266
CNT_TRAN_SUP_TENDENCY3M    -0.524888
REST_DYNAMIC_PAYM_3M        2.071528
CNT_TRAN_ATM_TENDENCY3M    -0.892280
CLNT_SETUP_TENOR           -0.092612
REST_DYNAMIC_CC_1M         -0.083145
PACK_101                   -0.072293
PACK_102                   -0.700611
PACK_103                   -0.275304
PACK_104                   -0.139191
PACK_105                    2.624803
PACK_107                   -0.292075
PACK_108                   -0.002740
PACK_109                   -0.015380
PACK_301                   -0.109144
PACK_K01                   -0.526102
PACK_M01                   -0.004746
PACK_O01                   -0.406695
```

In this instance, while CR_PROD_CNT_PIL has the highest weight and is negative, there are multiple features which are affecting the model positively. As the prediction is false we can infer that the weights for the positive might've been calculated at a glance, however further investigation is required for an informed decision.

SHAP

SHAP (SHapley Additive exPlanations) is used to explain machine learning model output. Each feature's contribution to the model's prediction is indicated, providing both local and global interpretability. SHAP values, which are based on cooperative game theory, offer consistent and unambiguous explanations for why a model generates particular predictions, which helps with debugging and understanding of models in a variety of contexts.



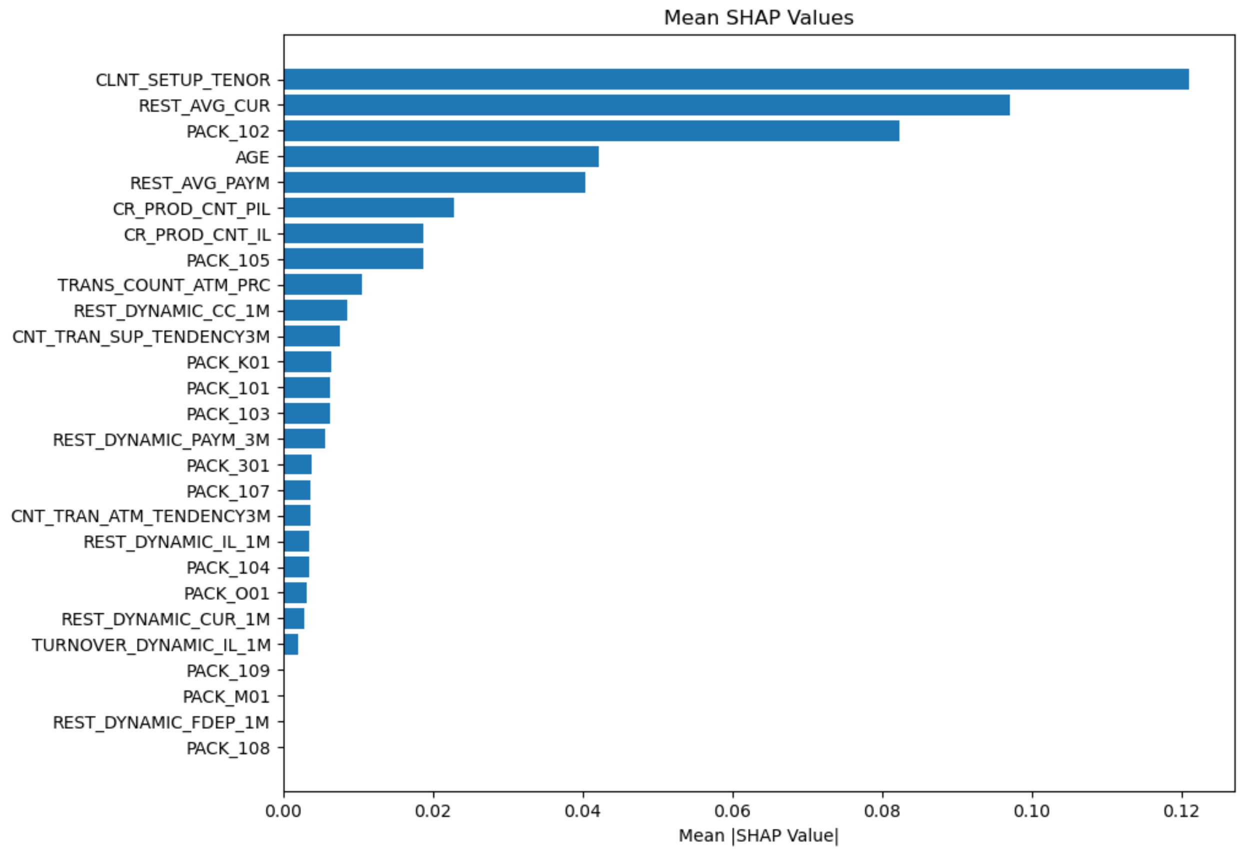
```
In [36]: import shap
         shap.initjs()

In [49]: X_train_sampled = shap.sample(X_train,100)

In [50]: explainer = shap.KernelExplainer(svc.predict_proba, X_train_sampled)

In [51]: X_test_sampled = shap.sample(X_test,100)

In [53]: shap_values = explainer.shap_values(X_test_sampled)

           0%|          | 0/100 [00:00<?, ?it/s]

In [54]: mean_shap_values = np.mean(np.abs(shap_values), axis=0)
         feature_names = X_train_sampled.columns

         sorted_indices = np.argsort(np.sum(np.abs(mean_shap_values), axis=1))
         sorted_mean_shap_values = mean_shap_values[sorted_indices]
         sorted_feature_names = np.array(feature_names)[sorted_indices]

         plt.figure(figsize=(10, 8))
         plt.barh(range(len(sorted_feature_names)), sorted_mean_shap_values[:, 0], align='center')
         plt.yticks(range(len(sorted_feature_names)), sorted_feature_names)
         plt.xlabel('Mean |SHAP Value|')
         plt.title('Mean SHAP Values')
         plt.show()
```

Global Feature Importance

Global feature importance with SHAP refers to the overall contribution of each feature across the entire dataset in influencing the model's predictions. When analyzing feature importance, SHAP (SHapley Additive exPlanations) values offer a thorough insight by taking feature interactions and their effects on model outcomes into account. These values can be analyzed to see which features have the biggest impact on the model's predictions. This information can then be used to modify the model, choose features, and gain a better understanding of the underlying patterns in the data.

In our dataset, We have used a subsample of 100 data points to calculate the mean SHAP values. We are taking the absolute of the said SHAP values to calculate the mean SHAP, so that negative and positive values don't cancel each other.

By the following diagram we can infer that features like CLNT_SETUP_TENOR(months being a customer),REST_AVG_CUR (Average current account balances),PACK_102(Clients with 102 as severance package) and AGE have the most weightage in the model and more likely to decide the outcome/ prediction of the model.


Mean SHAP Values
CLNT_SETUP_TENOR
REST_AVG_CUR
PACK_102
AGE
REST_AVG_PAYM
CR_PROD_CNT_PIL
CR_PROD_CNT_IL
PACK_105
TRANS_COUNT_ATM_PRC
REST_DYNAMIC_CC_1M
CNT_TRAN_SUP_TENDENCY3M
PACK_K01
PACK_101
PACK_103
REST_DYNAMIC_PAYM_3M
PACK_301
PACK_107
CNT_TRAN_ATM_TENDENCY3M
REST_DYNAMIC_IL_1M
PACK_104
PACK_O01
REST_DYNAMIC_CUR_1M
TURNOVER_DYNAMIC_IL_1M
PACK_109
PACK_M01
REST_DYNAMIC_FDEP_1M
PACK_108
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Mean |SHAP Value|

Findings/ Conclusion

Previously, results were not good as the AUC curve was below the middle line. Even after converting age from months to year, there was no change in error. We checked the PACK column as it was impacting positive/negative classes for CHURN using Lime. The PACK column was removed completely to check for the values that could be biasing the model. Error was the same.

After seeing the data, the PACK class had very imbalanced data. So, we stratified our train test validation creation. We did get better results in comparison to previous ROC AUC. We saw that there are few outliers in different columns. Treating the outliers did not seem to improve the model. Though we got a better ROC Curve, but yet precision is not good. We are slowly progressing

MECE Table

Code Quality	Mohamed Maaz: Did the code quality and comments Aman Nain: Did the code quality
Sensitivity Analysis	Rodrigo Velazquez: Did the Sensitivity Analysis Aman Nain: Helped with the Analysis
Local feature importance	Leandro Sartini: Performed The local feature importance using Lime Mohamed Maaz: Helped with the analysis
Global feature importance	Saraka Sahaswin: Did the global Feature Importance Prakash Pun: Did the global feature importance Bibek Ranjit: Did the feature importance
Model Tuning	Leandro Sartini: Did the Model Tuning Rodrigo Velazquez Did the Model Tuning
Report	Bibek Ranjit: Did the Report Prakash Pun: Did the Report Saraka Sahaswin: Did the report