Default Prediction

Predict if a customer will default in the future

Team#2 An Lee, Nini Lin

Data Overview

Data Description

- Features are anonymized and normalized, and fall into the following general categories:

  D_* = Delinquency variables

  B_* = Balance variables

  S_* = Spend variables

  R_* = Risk variables

  P_* = Payment variables

- The following features are categorical:

  ['B_30', 'B_38', 'D_114', 'D_116', 'D_117', 'D_120', 'D_126', 'D_63', 'D_64', 'D_66', 'D_68']

Data Description

Files

- train_data.csv (16.39GB) : training data with multiple statement dates per customer_ID
- train_labels.csv (30.75MB) : target label for each customer_ID

Data Exploration

Reduce Data Size

Reduce columns to the least size

- CustomerID: from object to int64
- Categorical: from object to int8 (all categorical features have up to 8 category)
- Numerical: from float64 to float32, which is 4 bytes per row
- S_2 Column: from object to datetime, which is only 4 bytes

```
]:
def convert_train_parquet(input_path, output_path, chunksize = 15000):


    train_label_tmp = pd.read_csv("../input/amex-default-prediction/train_labels.csv")
    pq_writer = None

    for idx, df_chunk in enumerate(pd.read_csv(input_path, chunksize=chunksize)):
        print(f"id: {idx} Chunk size {df_chunk.shape}")

        df_chunk[cont_vars] = df_chunk[cont_vars].astype('float32')

        # merge a new column 'target'
        df_chunk = df_chunk.merge(train_label_tmp, left_on='customer_ID', right_on='customer_ID')

        # change the hash string to integer.
        df_chunk.customer_ID = df_chunk.customer_ID.apply(lambda x: int(x[-16:],16)).astype('int64')

        #Convert catwgorical data to int.

        df_chunk['B_30'] = convert_to_int(floorify_frac(df_chunk['B_30']))
        df_chunk['B_38'] = convert_to_int(floorify_frac(df_chunk['B_38']))
        df_chunk['D_66'] = convert_to_int(floorify_frac(df_chunk['D_66']))
        df_chunk['D_68'] = convert_to_int(floorify_frac(df_chunk['D_68']))
        df_chunk['D_114'] = convert_to_int(floorify_frac(df_chunk['D_114']))
        df_chunk['D_116'] = convert_to_int(floorify_frac(df_chunk['D_116']))
        df_chunk['D_117'] = convert_to_int(floorify_frac(df_chunk['D_117'])+1)
        df_chunk['D_120'] = convert_to_int(floorify_frac(df_chunk['D_120']))
        df_chunk['D_126'] = convert_to_int(floorify_frac(df_chunk['D_126'])+1)
        df_chunk['D_63'] = df_chunk['D_63'].apply(lambda t: {'CR':0, 'XZ':1, 'XM':2, 'CO':3, 'CL':4, 'XL':5}[t]).astype(np.int8)
        df_chunk['D_64'] = df_chunk['D_64'].apply(lambda t: {np.nan:-1, 'O':0, '-1':1, 'R':2, 'U':3}[t]).astype(np.int8)

        df_chunk['S_2'] = pd.to_datetime(df_chunk['S_2'])

        table = pa.Table.from_pandas(df_chunk)
        if idx == 0:
            pq_writer = pq.ParquetWriter(output_path, table.schema, compression = 'snappy')

        pq_writer.write_table(table)

        # Removing current chunk from meory to free up memory
        del df_chunk
        del table
        gc.collect()

    if pq_writer:
        pq_writer.close()
    gc.collect()
```

Original Dtype

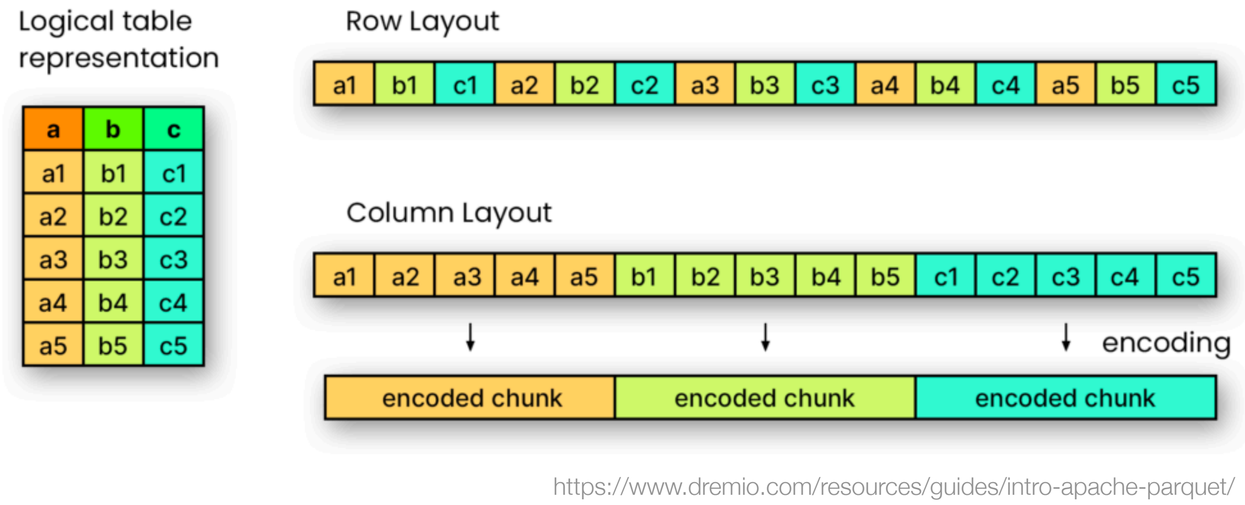
```
Data columns (total 190 columns):
 #    Column       Non-Null Count  Dtype
---   ------       --------------  -----
 0    customer_ID  60 non-null     object
 1    S_2          60 non-null     object
 2    P_2          60 non-null     float64
 3    D_39         60 non-null     float64
 4    B_1          60 non-null     float64
 5    B_2          60 non-null     float64
 6    R_1          60 non-null     float64
 7    S_3          43 non-null     float64
 8    D_41         60 non-null     float64
 9    B_3          60 non-null     float64
 10   D_42         0 non-null      float64
 11   D_43         28 non-null     float64
 12   D_44         60 non-null     float64
 13   B_4          60 non-null     float64
 14   D_45         60 non-null     float64
 15   B_5          60 non-null     float64
 16   R_2          60 non-null     float64
 17   D_46         60 non-null     float64
 18   D_47         60 non-null     float64
 19   D_48         53 non-null     float64
 20   D_49         0 non-null      float64
 21   B_6          60 non-null     float64
 22   B_7          60 non-null     float64
 23   B_8          60 non-null     float64
 24   D_50         34 non-null     float64
 25   D_51         60 non-null     float64
 26   B_9          60 non-null     float64
 27   R_3          60 non-null     float64
 28   D_52         60 non-null     float64
 29   P_3          60 non-null     float64
 30   B_10         60 non-null     float64
 31   D_53         13 non-null     float64
 32   S_5          60 non-null     float64
 33   B_11         60 non-null     float64
 34   S_6          60 non-null     float64
 35   D_54         60 non-null     float64
 36   R_4          60 non-null     float64
 37   S_7          43 non-null     float64
 38   B_12         60 non-null     float64
 39   S_8          60 non-null     float64
 40   D_55         60 non-null     float64
 41   D_56         55 non-null     float64
 42   B_13         60 non-null     float64
 43   R_5          60 non-null     float64
 44   D_58         60 non-null     float64
 45   S_9          30 non-null     float64
 46   B_14         60 non-null     float64
 47   D_59         60 non-null     float64
 48   D_60         60 non-null     float64
 49   D_61         54 non-null     float64
 50   B_15         60 non-null     float64
 51   S_11         60 non-null     float64
 52   D_62         50 non-null     float64
 53   D_63         60 non-null     object
 54   D_64         60 non-null     object
 55   D_65         60 non-null     float64
```

Converted Dtype

```
Data columns (total 191 columns):
 #    Column       Non-Null Count    Dtype
---   ------       --------------    -----
 0    customer_ID  5531451 non-null  int64
 1    S_2          5531451 non-null  datetime64[ns]
 2    P_2          5485466 non-null  float32
 3    D_39         5531451 non-null  float32
 4    B_1          5531451 non-null  float32
 5    B_2          5529435 non-null  float32
 6    R_1          5531451 non-null  float32
 7    S_3          4510907 non-null  float32
 8    D_41         5529435 non-null  float32
 9    B_3          5529435 non-null  float32
 10   D_42         791314 non-null   float32
 11   D_43         3873055 non-null  float32
 12   D_44         5257132 non-null  float32
 13   B_4          5531451 non-null  float32
 14   D_45         5529434 non-null  float32
 15   B_5          5531451 non-null  float32
 16   R_2          5531451 non-null  float32
 17   D_46         4319752 non-null  float32
 18   D_47         5531451 non-null  float32
 19   D_48         4812726 non-null  float32
 20   D_49         545534 non-null   float32
 21   B_6          5531218 non-null  float32
 22   B_7          5531451 non-null  float32
 23   B_8          5509183 non-null  float32
 24   D_50         2389049 non-null  float32
 25   D_51         5531451 non-null  float32
 26   B_9          5531451 non-null  float32
 27   R_3          5531451 non-null  float32
 28   D_52         5501888 non-null  float32
 29   P_3          5229959 non-null  float32
 30   B_10         5531451 non-null  float32
 31   D_53         1446866 non-null  float32
 32   S_5          5531451 non-null  float32
 33   B_11         5531451 non-null  float32
 34   S_6          5531451 non-null  float32
 35   D_54         5529435 non-null  float32
 36   R_4          5531451 non-null  float32
 37   S_7          4510907 non-null  float32
 38   B_12         5531451 non-null  float32
 39   S_8          5531451 non-null  float32
 40   D_55         5346648 non-null  float32
 41   D_56         2540508 non-null  float32
 42   B_13         5481932 non-null  float32
 43   R_5          5531451 non-null  float32
 44   D_58         5531451 non-null  float32
 45   S_9          2597808 non-null  float32
 46   B_14         5531451 non-null  float32
 47   D_59         5424726 non-null  float32
 48   D_60         5531451 non-null  float32
 49   D_61         4933399 non-null  float32
 50   B_15         5524528 non-null  float32
 51   S_11         5531451 non-null  float32
 52   D_62         4773290 non-null  float32
 53   D_63         5531451 non-null  int8
 54   D_64         5531451 non-null  int8
 55   D_65         5531451 non-null  float32
```

Parquet Format

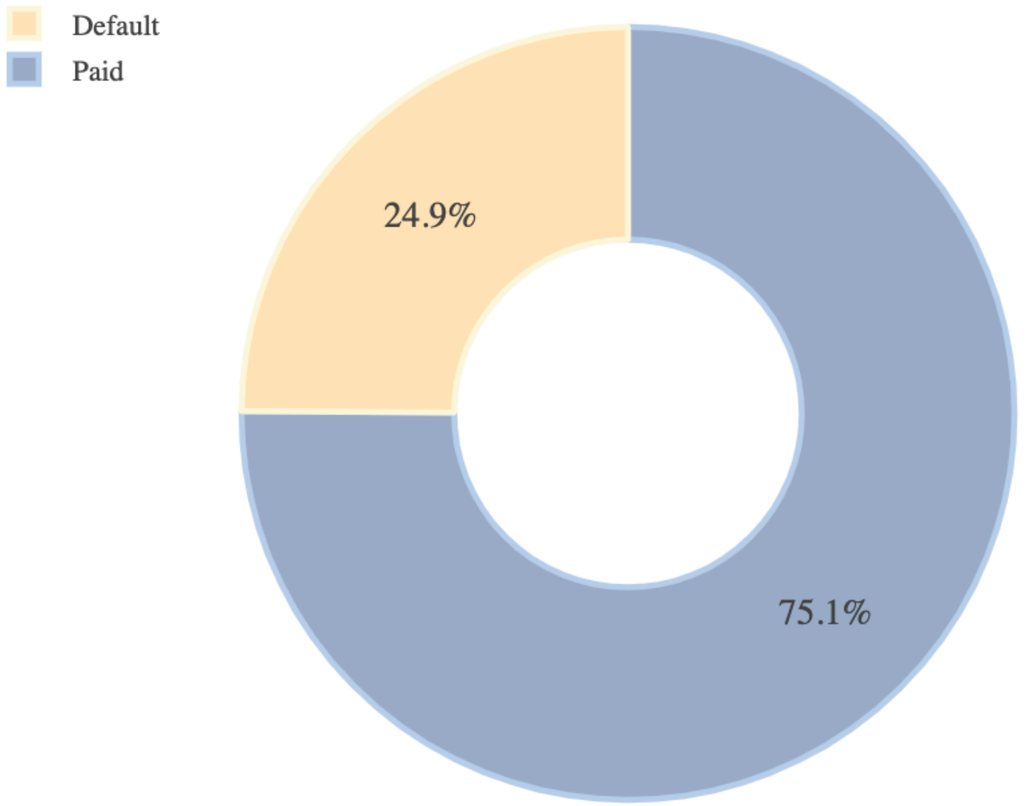
- Column-based format: files are organized by column, rather than by row
- Save storage space and speed up analytics queries
- Provide efficient data compression to handle complex data in bulk



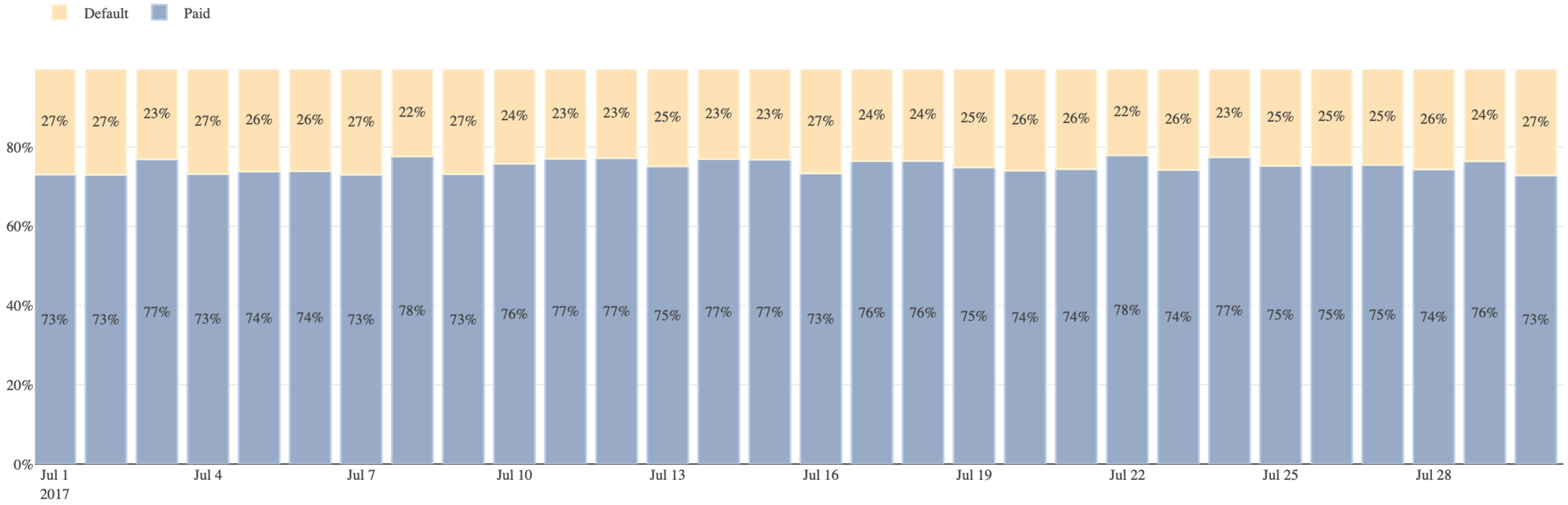
https://www.dremio.com/resources/guides/intro-apache-parquet/

Shape of Data

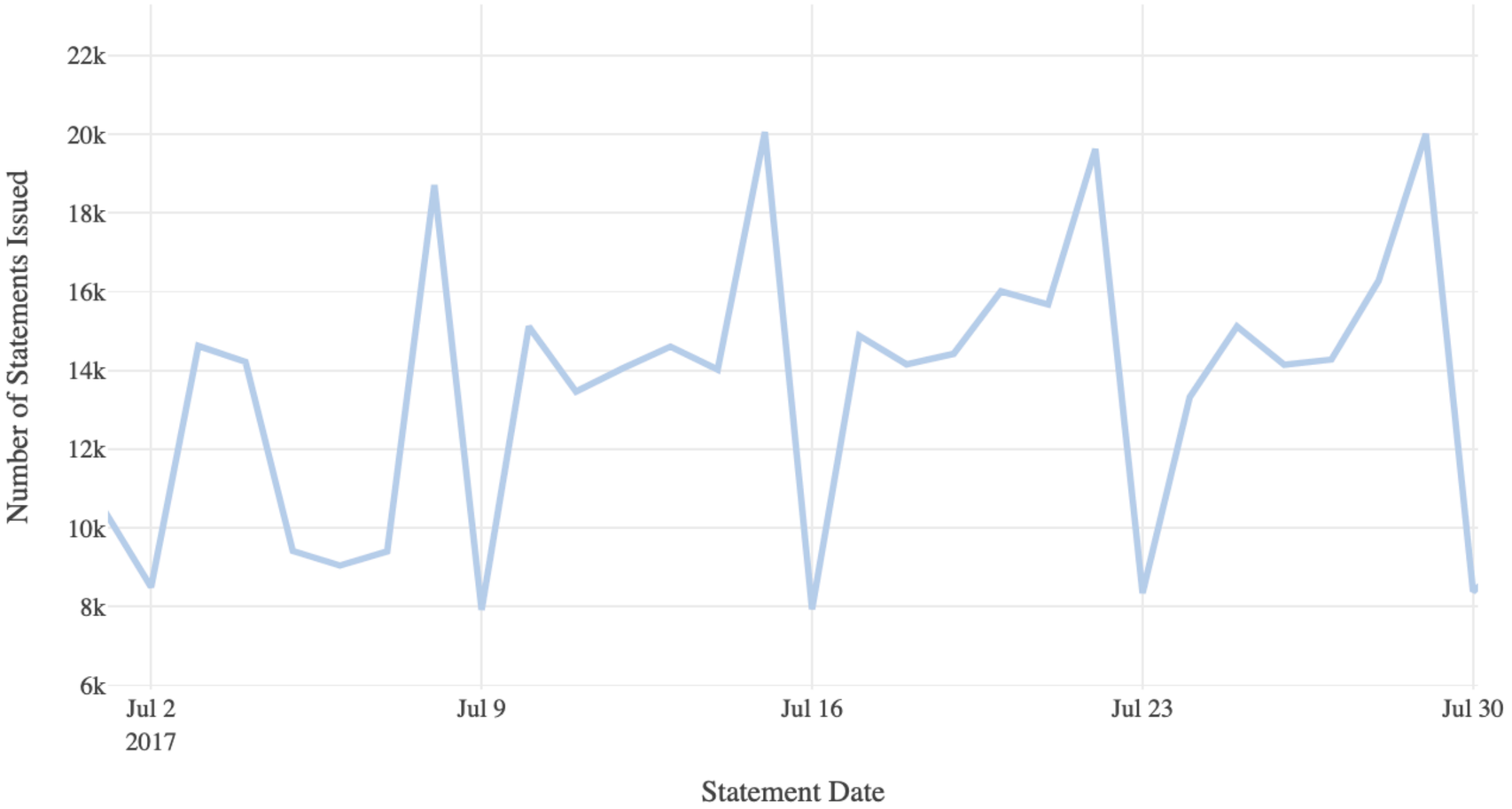
Target Distribution



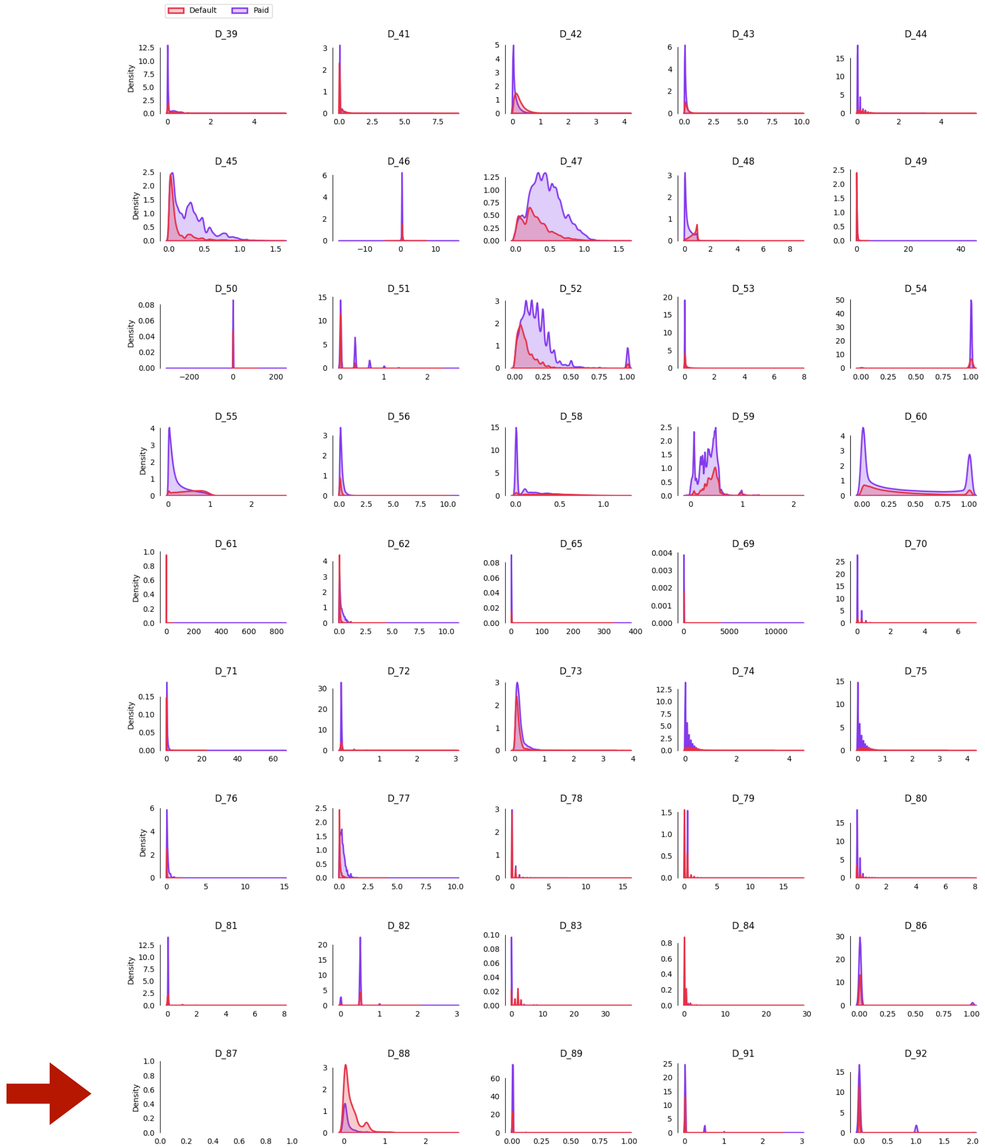
Distribution of Target by Day

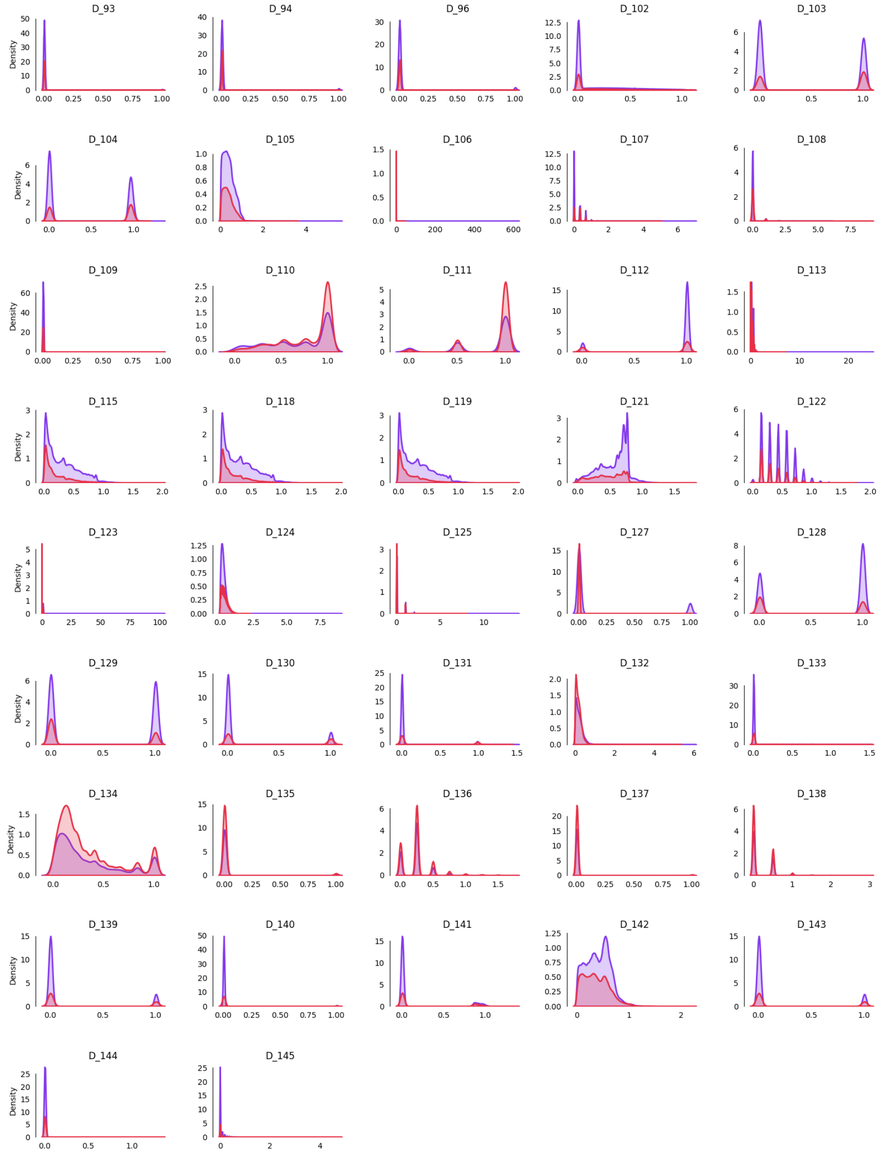


Frequency of Statements by Day



Distribution of Delinquency Feature





Correlation between Delinquency features

Pairs that are **positively** correlated at 1.0:
(D_62, D_77), (D_103, D_104), (D_139, D_141), (D_139, D_143), (D_141, D_143)

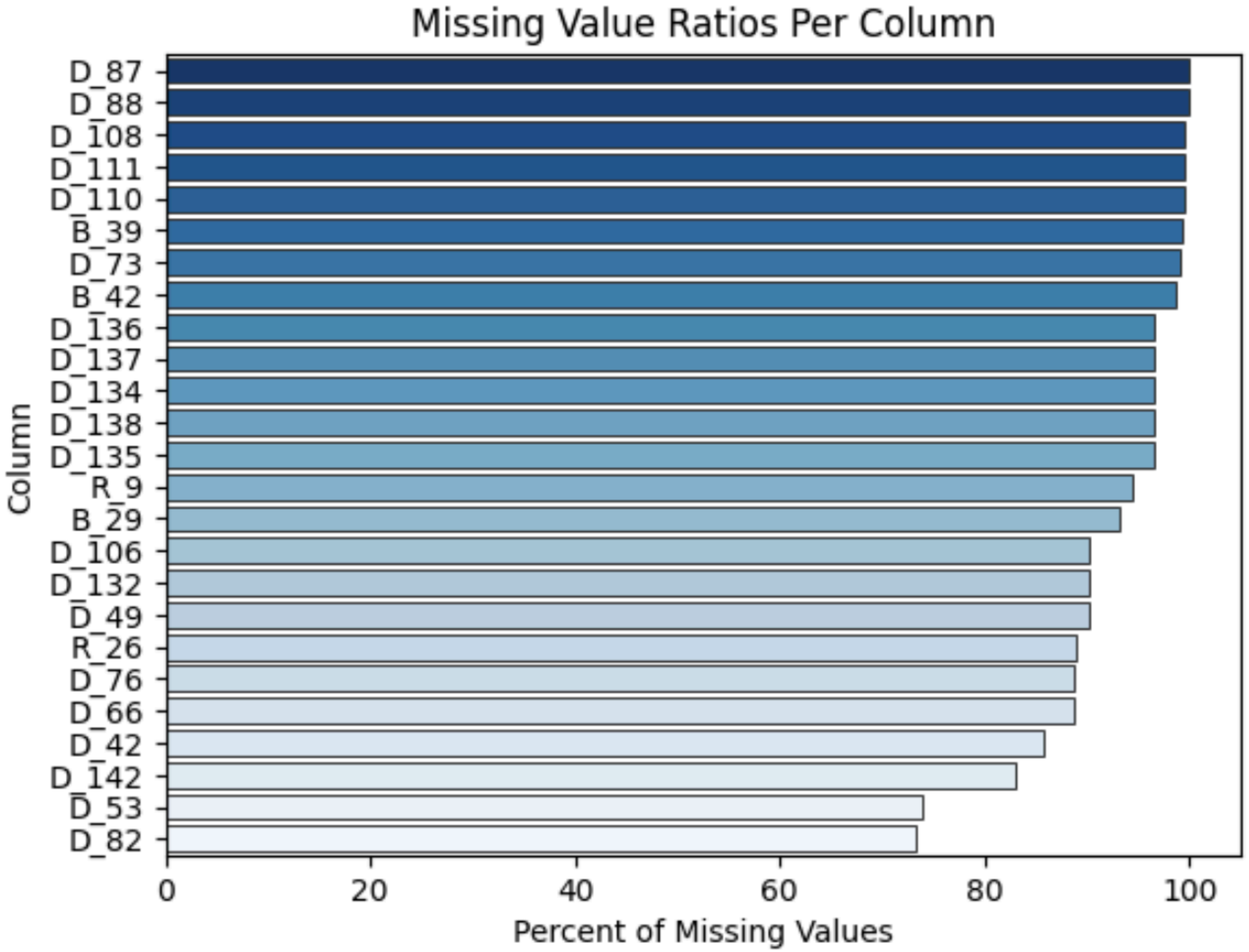
Missing correlations, particularly in D_87, D_111, D_110, D_73, due to their large amount of missing values in the data.

Pairs that are **negatively** correlated:
(D_41, D_54), (D_41, D_112), (D_113, D_115)

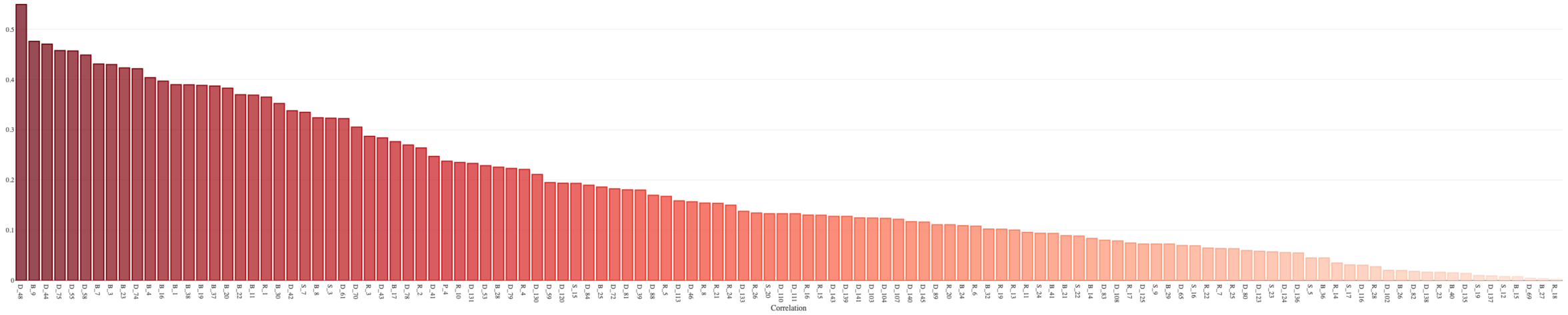
Missing Value

Solution

- Dropping
- Impution: replace with values

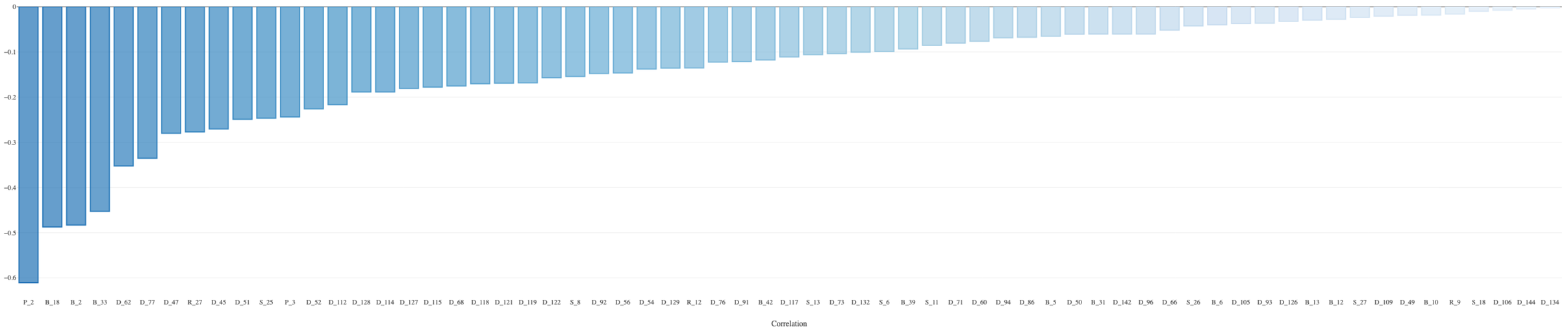


Correlation b/w target and features



Delinquency 48 is the most **positively** correlated overall at 0.61.

Correlation b/w target and features



Payment 2 is the most **negatively** correlated with the probability of defaulting with a correlation of -0.67.

Data Preparation

Handling Imbalanced Data

Evaluation Metric

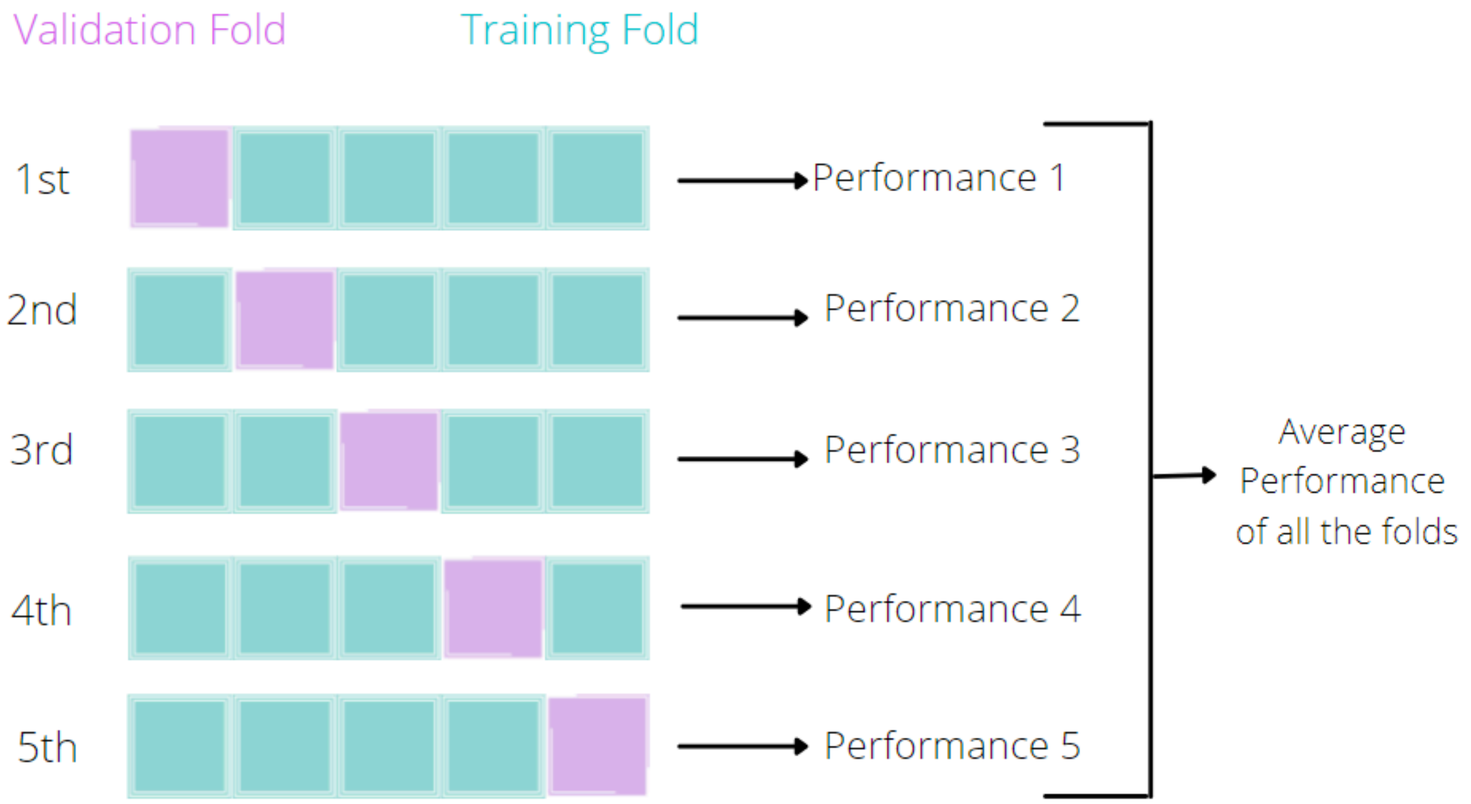
- Negative class (default) has been subsampled for this dataset at 5%, and thus receives a 20x weighting in the scoring metric.

- The evaluation metric M is the mean of two measures of rank ordering: Normalized Gini Coefficient, G, and default rate captured at 4%, D.

$$M = 0.5 * (G+D)$$

K-fold Cross-Validation

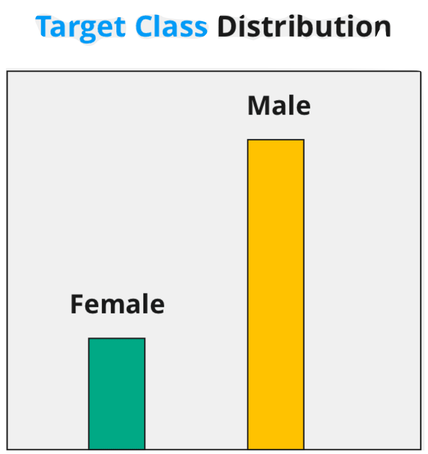
- A resampling procedure used to evaluate machine learning models on a limited data sample which is split into K groups with a uniform probability distribution

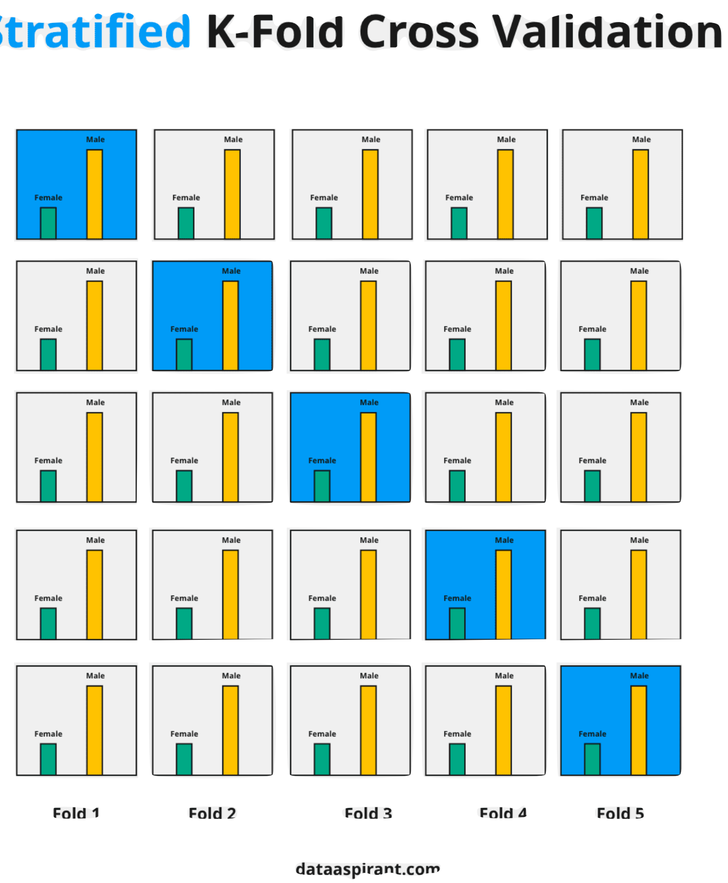


https://towardsai.net/p/machine-learning/k-fold-cross-validation-for-machine-learning-models

Stratified K-fold Cross-Validation

- Same as the regular K-fold method, but it preserves the imbalanced class distribution in the training and testing dataset of each fold





https://dataaspirant.com/8-stratified-k-fold-cross-validation/

Data Modeling

Random Forest

Evaluation Score:

- Baseline: 0.7552
- K-Fold: 0.984

In [29]:
```
dtc = RandomForestClassifier(
    n_estimators = 200, random_state=0
#     max_depth = 8
)


dtc.fit(X_train, y_train)
```

Out[29]:
```
RandomForestClassifier(n_estimators=200, random_state=0)
```

In [31]:
```
preds = dtc.predict(X_test)
# Compute accuracy
accuracy = accuracy_score(y_test, preds)
print(f'accuracy: {accuracy: .2%}')

y_pred = pd.DataFrame(y_test.copy(deep=True))
y_pred = y_pred.rename(columns={'target':'prediction'})
# preds_prob = xg_cl.predict_proba(X_test)[:,1]
y_pred['prediction'] = dtc.predict_proba(X_test)[:,1]
```

```
accuracy:  89.20%
```

In [32]:
```
print('Metric Evaluation Values\n')
# print(f'Numpy: {amex_metric(y_test.to_frame(), preds_prob_df)}')
print(f'Numpy: {amex_metric(y_test.to_frame(), y_pred)}')
```

```
Metric Evaluation Values

Numpy: 0.7551540038830189
```

In [ ]:

LightGBM

Evaluation Score

- Baseline: 0.7832
- K-Fold: 0.7866
- Stratified K-Fold: 0.7867

[8]:
```
# Create the arrays for features and the target: X, y
features = [x for x in train.columns.values if x not in ['customer_ID', 'target', 'S_2']]
X, y = train[features], train['target']

# Create the training and test datasets
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size = 0.2,
                                                    random_state=100,
                                                    stratify=y)
```

[9]:
```
clf = LGBMClassifier(
    n_estimators=50000,
    device='gpu',
    random_state=0,
    extra_trees=True
)
```

[10]:
```
%%time
clf.fit(
    X_train, y_train,
    eval_set=[(X_test,y_test)],
    callbacks=[early_stopping(50), log_evaluation(0)]
)
```

```
Training until validation scores don't improve for 50 rounds
Early stopping, best iteration is:
[278]   valid_0's binary_logloss: 0.223231
CPU times: user 56.7 s, sys: 1.05 s, total: 57.7 s
Wall time: 33.7 s
```

[10…]
```
LGBMClassifier(device='gpu', extra_trees=True, n_estimators=50000,
               random_state=0)
```

[14]:
```
%%time
y_test = pd.DataFrame(y_test)
amex_metric(y_test, y_pred)
```

```
CPU times: user 137 ms, sys: 908 µs, total: 138 ms
Wall time: 138 ms
```

[14…]
```
0.7832347768676333
```

XGBoost

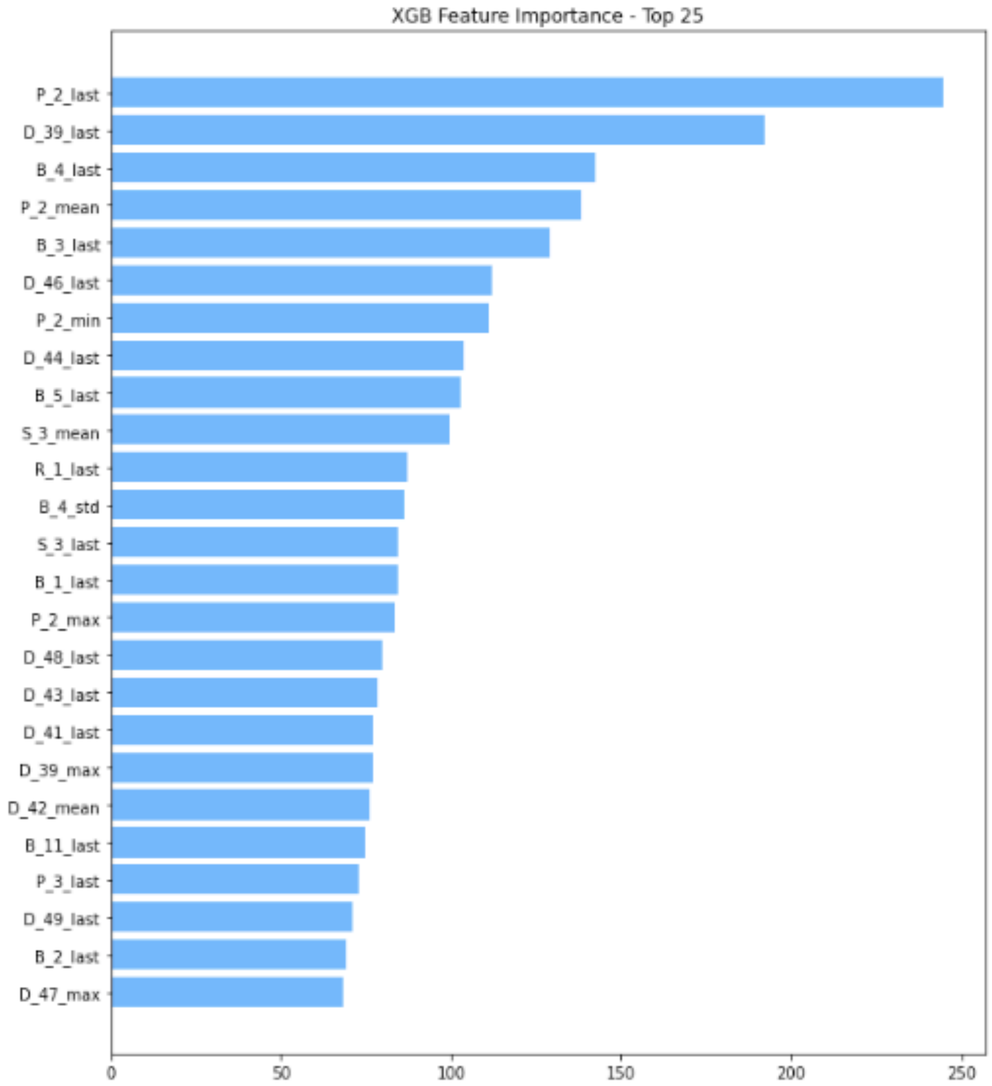
Evaluation Score

- Baseline: 0.7774
- K-Fold: 0.7867
- Stratified K-Fold: 0.7869

Preprocessing

-Feature Engineering: Numerical, Categorical

- K-Fold: 0.791
- Stratified K-Fold: 0.791



“Thank you for listening”

– Team #2