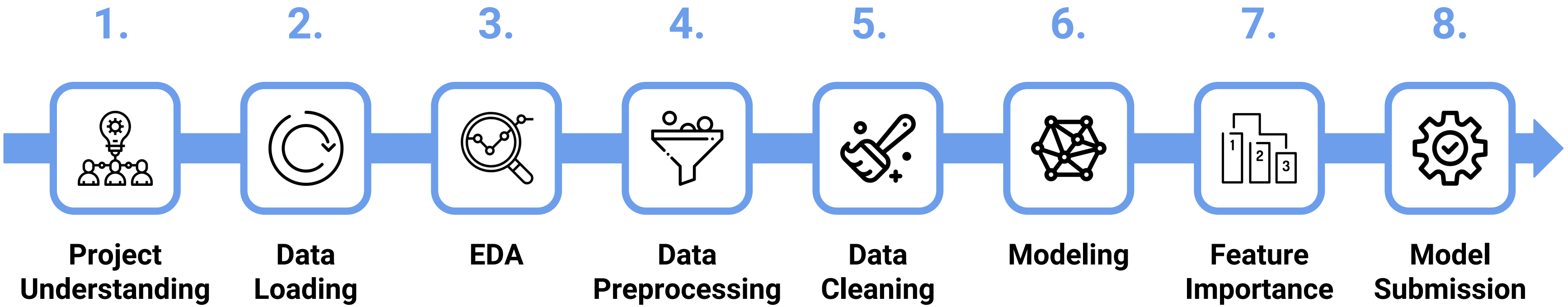
Elo Kaggle Challenge
- Final Results Group 6 -

Tarazali Ryskul	742820	s_ryskul18@stud.hwr-berlin.de
Sara Sommerfeld	353391	s_sommerfeld18@stud.hwr-berlin.de
Polina Voroshylova	743340	s_voroshylova18@stud.hwr-berlin.de

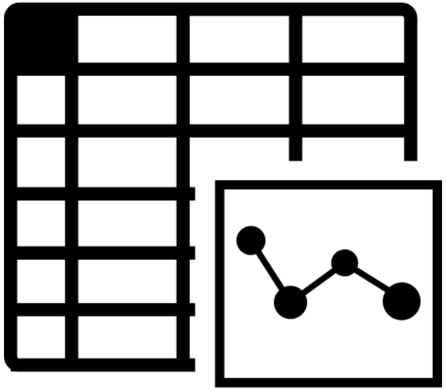
Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Our approach to solve the Elo Challenge



1. Project Understanding

Project Understanding: Data Overview and Challenge



train.csv	historical_transactions.csv
test.csv	merchants.csv
sample_submission.csv	new_merchant_transactions.csv

Target **customer loyalty**

2. Data Loading

Data Loading and Size Reduction

```
def reduce_mem_usage(df, verbose=True):
    numerics = ['int16', 'int32',
'int64', 'float16', 'float32', 'float64']
    start_mem = df.memory_usage().sum() /
1024**2
    for col in df.columns:
        col_type = df[col].dtypes
        if col_type in numerics:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] ==
'int':
                if c_min >
np.iinfo(np.int8).min and c_max <
np.iinfo(np.int8).max:
                    df[col] =
df[col].astype(np.int8)
  ...
```

```
Mem. usage decreased to  4.04 Mb (56.2%
reduction)
Mem. usage decreased to  2.24 Mb (52.5%
reduction)
Mem. usage decreased to 1749.11 Mb
(43.7% reduction)
Mem. usage decreased to 114.20 Mb
(45.5% reduction)
Mem. usage decreased to 30.32 Mb (46.0%
reduction)
```

3. Exploratory Data Analysis (EDA)

EDA: Train and Test Data

Test Data Sample

	first_active_month	card_id	feature_1	feature_2	feature_3
0	2017-04-01	C_ID_0ab67a22ab	3	3	1
1	2017-01-01	C_ID_130fd0cbdd	2	3	0
2	2017-08-01	C_ID_b709037bc5	5	1	1
3	2017-12-01	C_ID_d27d835a9f	2	1	0
4	2015-12-01	C_ID_2b5e3df5c2	5	1	1

Training Data Sample

	first_active_month	card_id	feature_1	feature_2	feature_3	target
0	2017-06-01	C_ID_92a2005557	5	2	1	-0.820312
1	2017-01-01	C_ID_3d0044924f	4	1	0	0.392822
2	2016-08-01	C_ID_d639edf6cd	2	2	0	0.687988
3	2017-09-01	C_ID_186d6a6901	4	3	0	0.142456
4	2017-11-01	C_ID_cdbd2c0db2	1	3	0	-0.159790

Target Variable

EDA: Merchant and Historical Transaction Data

Merchant Data Sample

	merchant_id	merchant_group_id	merchant_category_id	subsector_id	numerical_1	numerical_2	category_1
0	M_ID_838061e48c	8353	792	9	-0.057465	-0.057465	N
1	M_ID_9339d880ad	3184	840	20			
2	M_ID_e726bbae1e	447	690	1			
3	M_ID_a70e9c5f81	5026	792	9			
4	M_ID_64456c37ce	2228	222	21			

Historical Transactions Sample

	authorized_flag	card_id	city_id	category_1	installments	category_3	merchant_category_id	merchant_i
0	Y	C_ID_4e6213e9bc	88	N	0	A	80	M_ID_e02(
1	Y	C_ID_4e6213e9bc	88	N	0	A	367	M_ID_86ec
2	Y	C_ID_4e6213e9bc	88	N	0	A	80	M_ID_979e
3	Y	C_ID_4e6213e9bc	88	N	0	A	560	M_ID_e6d5
4	Y	C_ID_4e6213e9bc	88	N	0	A	80	M_ID_e02(

Missing Values Check: Train Data

```
print('Train Data');display(missing_values(df_train))
print('Test Data');display(missing_values(df_test))
print('New Merchants');display(missing_values(
df_new_merchant_trans))
print('Historical Transactions');display(missing_values(
df_hist_trans))
print('Merchants');display(missing_values(df_merchants))
```

```
Train Data
Your data contains 6
columns and has 0
columns with missing
values
```

Missing Values: Overview of Analysis

Train Data	Test Data	New Merchants	Historical Transactions	Merchants
Train Data Your data contains 6 columns and has 0 columns with missing values	Test Data Your data contains 5 columns and has 1 column with missing values	New Merchants Your data contains 14 columns and has 3 columns with missing values	Historical Transactions Your data contains 14 columns and has 3 columns with missing values	Merchants Your data contains 22 columns and has 4 columns with missing values

Missing Values: New Merchants Example

Train Data	Test Data	New Merchants	Historical Transactions	Merchants
Train Data Your data contains 6 columns and has 0 columns with missing values	missing values	missing values	missing values	Merchants Your data contains 22 columns and has 4 columns with missing values

	Total Miss Values	% of miss values
category_2	111745	5.69
category_3	55922	2.85
merchant_id	26216	1.34

EDA: Feature Analysis

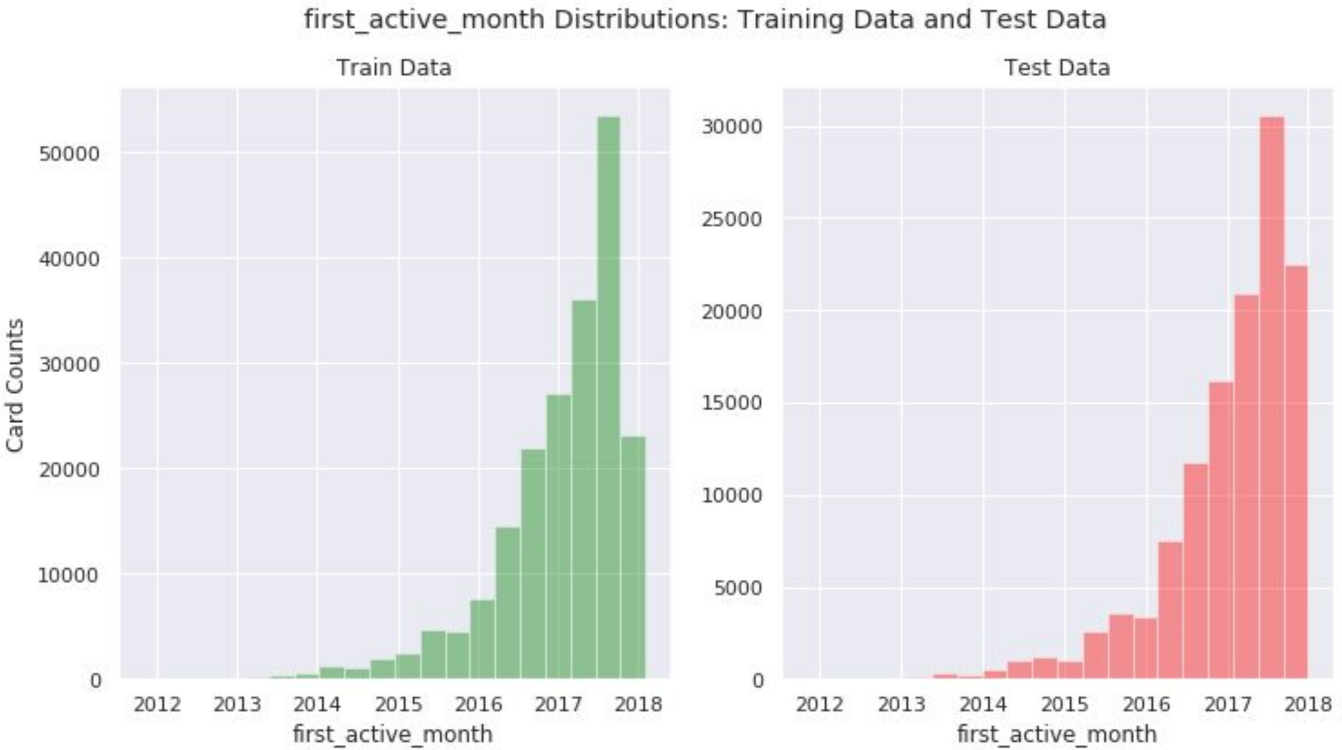
```
...
sns.distplot( df_train.feature_1,ax=axes[0], kde =
False, color = 'green', bins=10).set_title("Train Data")
sns.distplot( df_test.feature_1,ax=axes[1], kde = False,
color = 'red', bins=10).set_title("Test Data")
axes[0].set(ylabel='Card Counts')
...
plt.show()
```

EDA: Feature Analysis

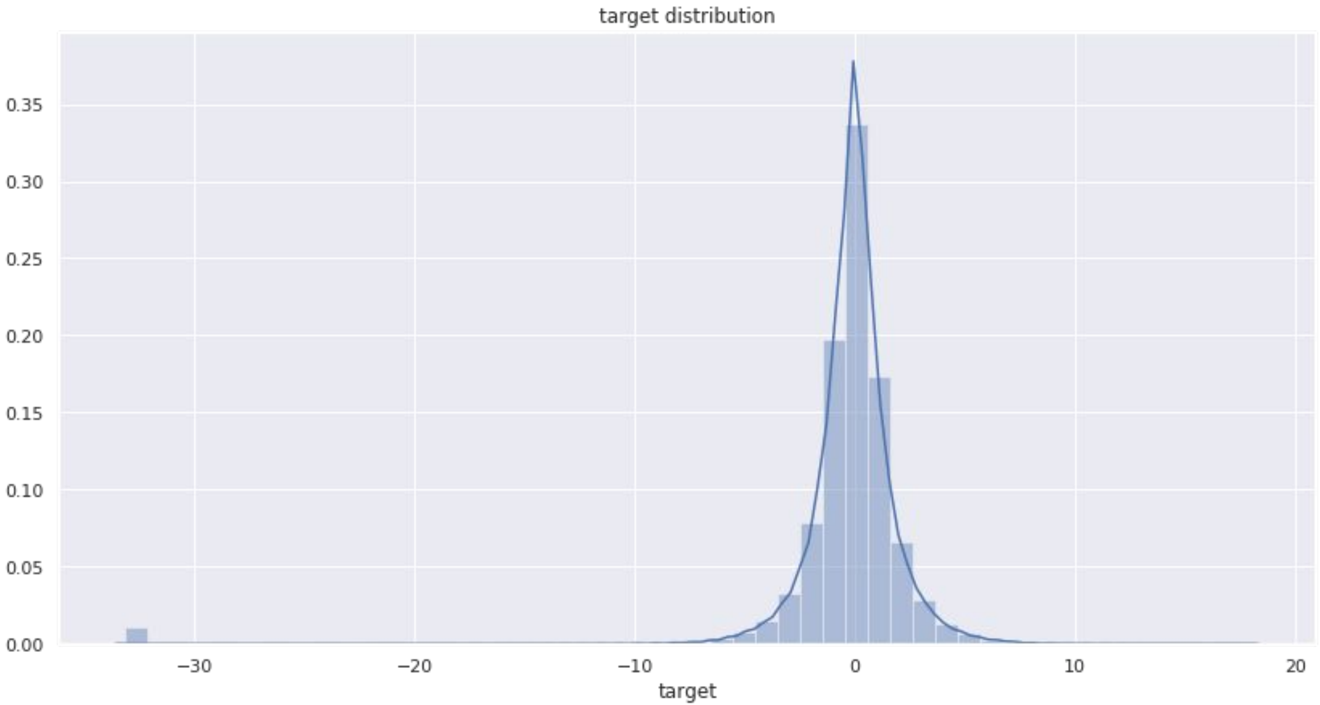
EDA: Feature Analysis

EDA: Feature Analysis

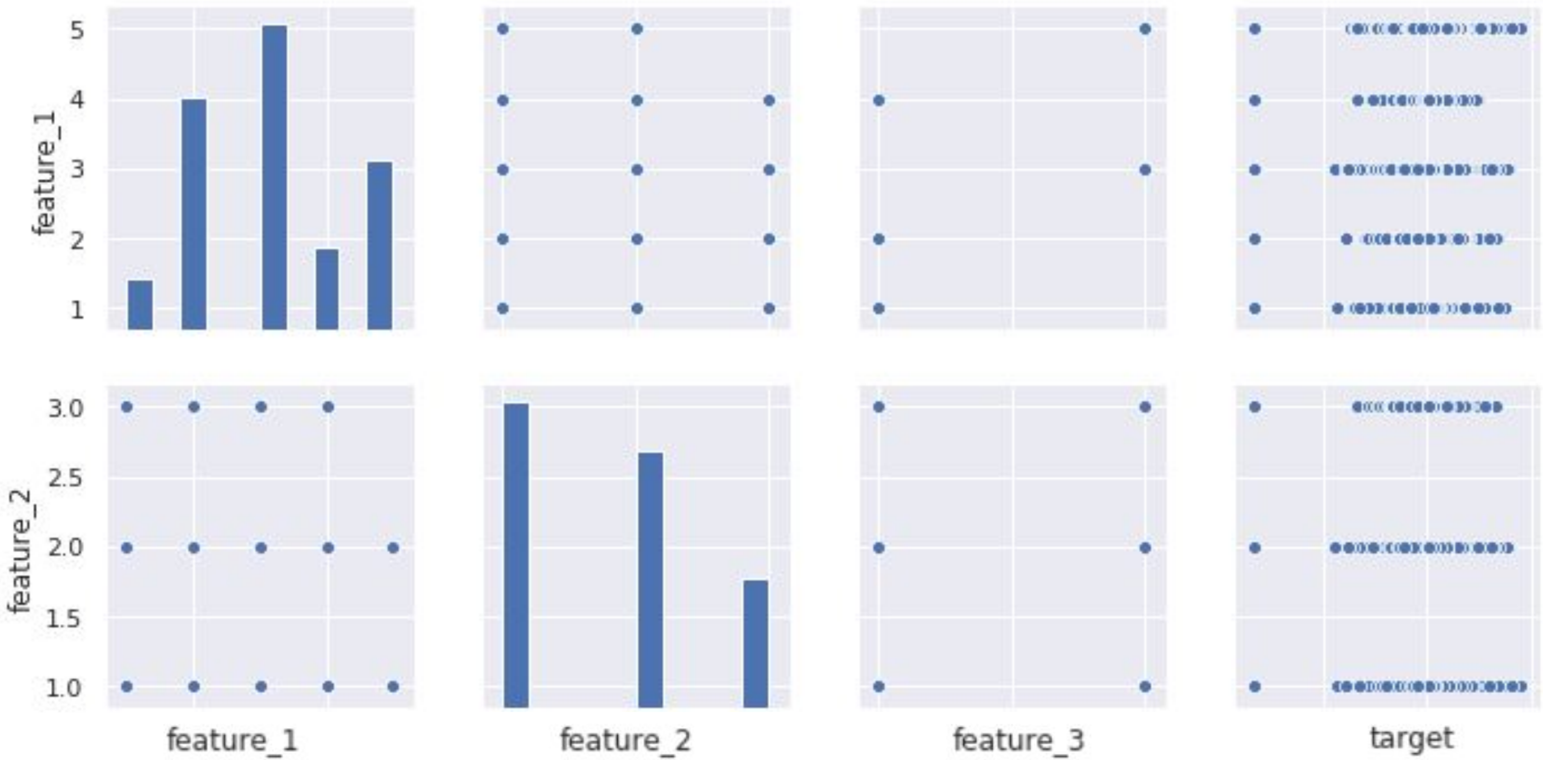
EDA: Feature Analysis



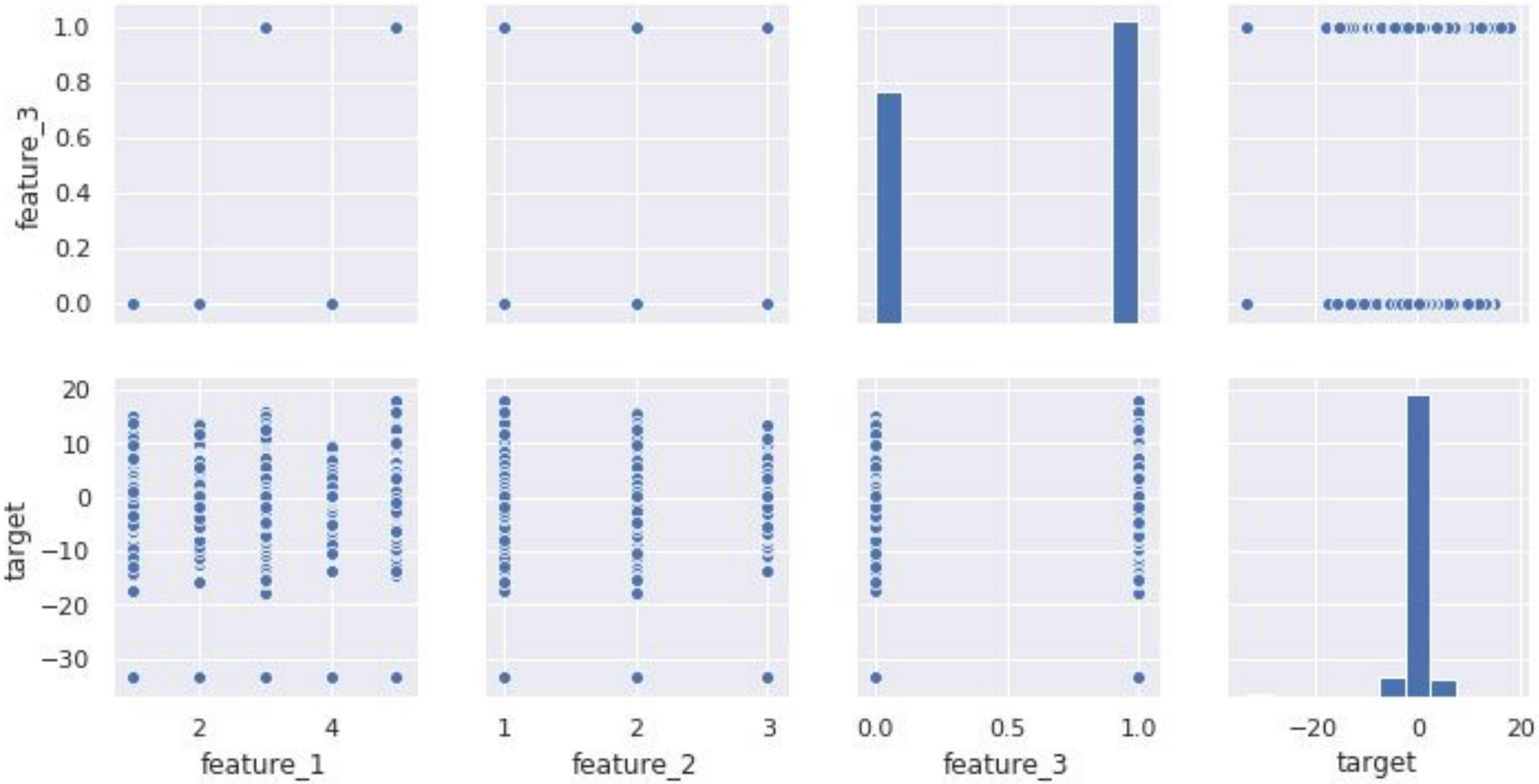
EDA: Distribution of Target Variable



EDA: Correlation of Features and Target Variable



EDA: Correlation of Features and Target Variable



Column Analysis: purchase_amount

```
df_hist_trans.purchase_amount.describe()
```

```
count     2.911236e+07
mean      6.134567e-02
std       1.123521e+03
min      -7.469078e-01
25%      -7.203559e-01
50%      -6.883495e-01
75%      -6.032543e-01
max       6.010604e+06
```

Column Analysis: purchase_amount

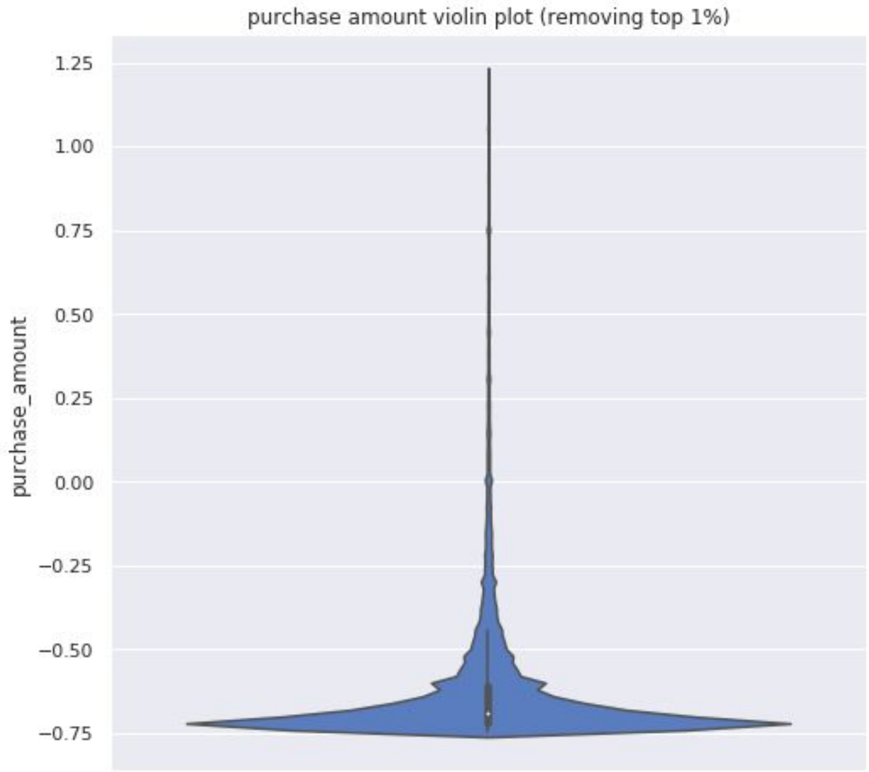
```
df_hist_trans.purchase_amount.describe()
```

```
count     2.911236e+07
mean      6.134567e-02
std       1.123521e+03
min      -7.469078e-01
25%      -7.203559e-01
50%      -6.883495e-01
75%      -6.032543e-01
max       6.010604e+06
```

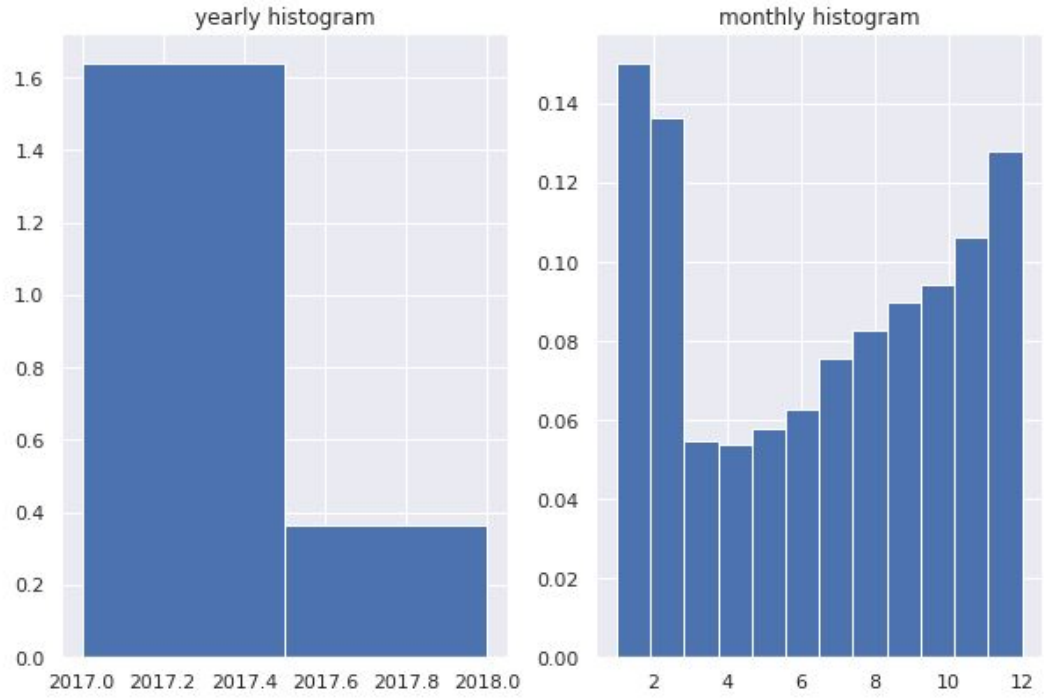
Column Analysis: purchase_amount

```
np.percentile(df_hist_trans
["purchase_amount"].values,q=99)
```

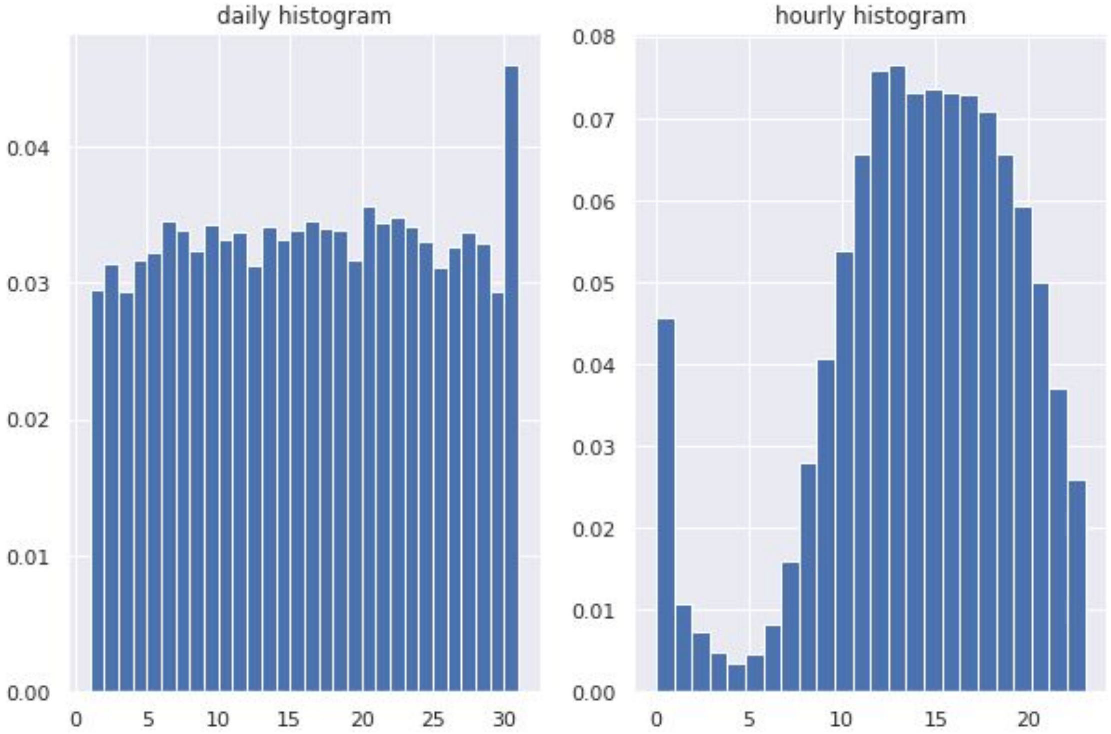
```
1.2208409547805337
```



Feature Analysis: Yearly and Monthly Histogram

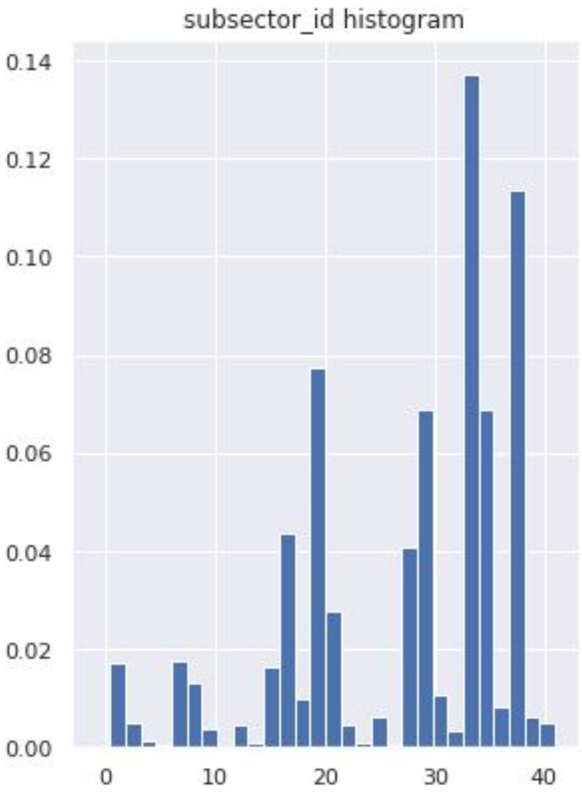


Feature Analysis: Daily and Hourly Histogram



Feature Analysis: state_id and city_id Histogram

Feature Analysis: subsector_id Histogram



4. Data Preprocessing

Data Preprocessing

```python
def aggregate_historical_transactions(history):
    ...
    agg_func = {
        'authorized_flag': ['sum', 'mean'],
        'merchant_id': ['nunique'],
        'city_id': ['nunique'],
        'state_id': ['nunique'],
        'purchase_amount': ['sum', 'median', 'max', 'min', 'std'],
        'installments': ['sum', 'median', 'max', 'min', 'std'],
        'purchase_date': [np.ptp],
       ...
        }
```

Data Preprocessing

```
def aggregate_historical_transactions(history):
```

	card_id	hist_transactions_count	hist_authorized_flag_sum	hist_authorized_flag_mean	hist_merchant_id_nunic
0	C_ID_00007093c1	149	114	0.765101	29
1	C_ID_0001238066	123	120	0.975610	65
2	C_ID_0001506ef0	66	62	0.939394	28
3	C_ID_0001793786	216	189	0.875000	119
4	C_ID_000183fdda	144	137	0.951389	73

```
        'purchase_date': [np.ptp],
       ...
        }
```

Data Preprocessing: elapsed_time Value Creation

```
from datetime import datetime
df_train['elapsed_time'] = (datetime(2018, 2, 1) - df_train['first_active_month']).dt.days
df_test['elapsed_time'] = (datetime(2018, 2, 1) - df_test['first_active_month']).dt.days
```

	first_active_month	card_id	feature_1	feature_2	feature_3	target	elapsed_time
0	2017-06-01	C_ID_92a2005557	5	2	1	-0.820312	245
1	2017-01-01	C_ID_3d0044924f	4	1	0	0.392822	396
2	2016-08-01	C_ID_d639edf6cd	2	2	0	0.687988	549
3	2017-09-01	C_ID_186d6a6901	4	3	0	0.142456	153
4	2017-11-01	C_ID_cdbd2c0db2	1	3	0	-0.159790	92

Data Preprocessing: Table Merging

```
df_train = pd.merge(df_train, new_history, on='card_id', how='left')
df_test = pd.merge(df_test, new_history, on='card_id', how='left')
df_train = pd.merge(df_train, new_merchants, on='card_id', how='left')
df_test = pd.merge(df_test, new_merchants, on='card_id', how='left')
```

```
df_train.shape, df_test.shape
```

```
((201917, 50), (123623, 49))
```

5. Data Cleaning

Data Cleaning

```
sns.distplot(df_train['new_purchase_amount_std'].dropna())
```

Data Cleaning

```
sns.distplot(df_train['new_purchase_amount_std'].dropna())
```

```
sns.distplot(df_train['new_purchase_amo
unt_std'].fillna(df_train['new_purchase
_amount_std'].mean()))
```

```
Your data contains 48 columns and has 0
columns with missing values
```

6. Modeling

Linear Models: Why we used Linear, Ridge and Lasso Regression

Linear Regression	Ridge Regression	Lasso Regression
Linear regression is the simplest form to be used for supervised learning particularly useful in order to predict quantitative responses.	Ridge regression shrinks the coefficients and it helps to reduce the model complexity and multi-collinearity.	Lasso regression not only helps in reducing overfitting but it can help us in the feature selection.

Non-linear Models: Why we used Trees and Random Forest

Decision Tree	Random Forest
■ Can handle both numerical and categorical values. ■ Perform well on large datasets. ■ Are extremely fast. ■ Resistant to outliers, hence require little data preprocessing.	■ Applied to build a number of decision trees on bootstrapped training samples. ■ Random subsets of features considered when the nodes are splitted.

Source: ISLR Book

Modeling: How we applied the selected Models

Linear Regression	Ridge Regression	Lasso Regression	Decision Tree Regressor	Random Forest
1. Cross validation 2. Fitting model	1. Tuning params with GridSearchCV 2. Fitting model with best params	1. Tuning params with GridSearchCV 2. Fitting model with best params	1. Fitting model with our own parameter values 2. Tuning parameters with GridSearchCV 3. Fitting model with best params	1. Fitting model with own params values 2. Fitting model with best params tuned by RandomizedSearchCV 3. Fitting model with best params tuned by RandomizedSearchCV

Results of Modeling: Comparison of RMSE

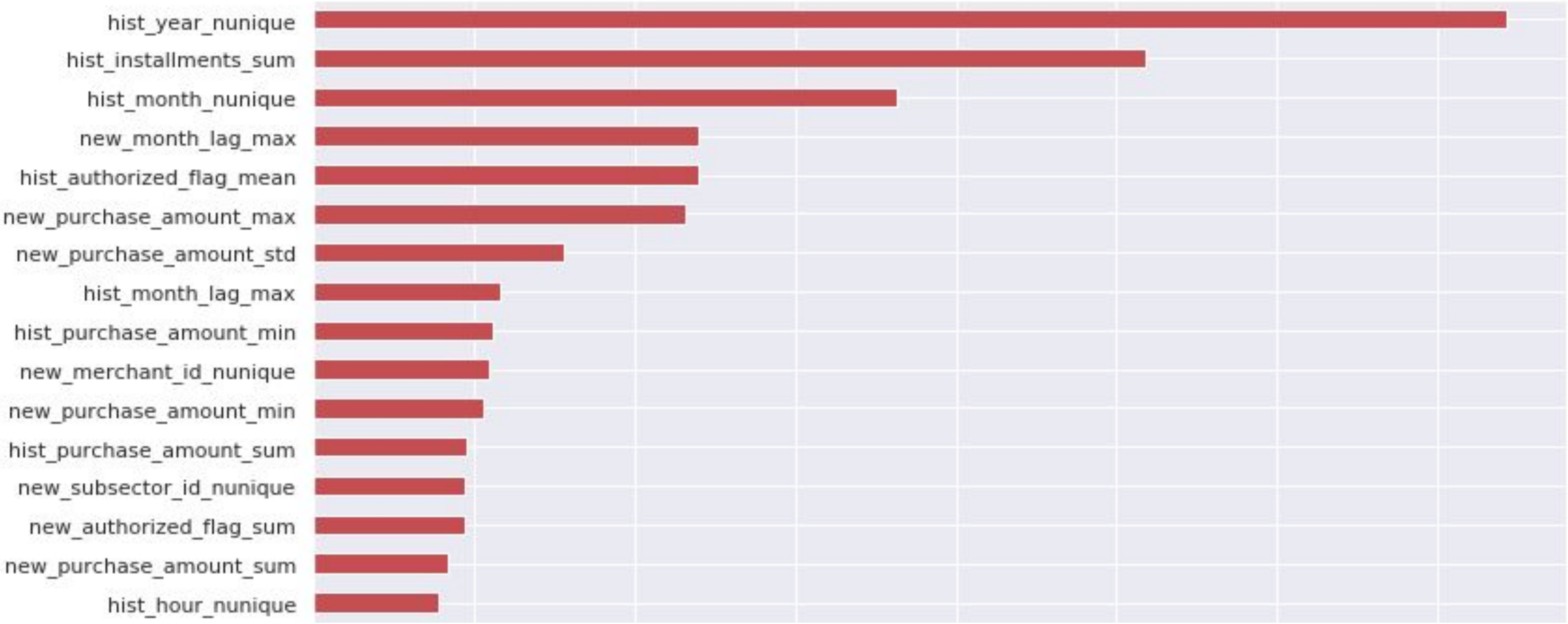
Linear Regression	Ridge Regression	Lasso Regression
RMSE: 3.7718	RMSE: 3.7983	RMSE: 3.7987

Decision Tree Regressor	Random Forest
RMSE: 3.7638	RMSE: 3.7419

Best RMSE

7. Feature Importance

Feature Importance: Overview



8. Model Submission and Conclusion

Our main Challenges during the Project

Main Challenges

Replacement of missing values

Some columns could not be replaced with the function of mean/median/min.

New feature creation

First time to create new features to fit model with difficult identification of necessary features.

Application of best models

Opportunity to apply knowledge from class while choosing the best models for the challenge.

Reduction of memory

Large dataset had to be reduced in size in order to not “crash” the server constantly.

Result and Conclusion

Best Model	RANDOM FOREST
Result	RMSE: 3.759
Leaderboard	2116 new DW_Project 3.759 11 5m Your Best Entry ↑

→ 2116 out of 3141 participants (status 28th January 2019)

THANK YOU