Nubank data challenge - Fraud Analysis

by Adriano Freitas

```
%%capture

""" Useful notebook definitions

Some usefull notebook definitions, like plots color scheme
and cell behavior were extracted to another notebook just
for a cleaner view
"""
%run ./utils.ipynb

# n_cores = cpu_count()

default_color = 'purple'
# default_light_color = 'white'
# default_dark_color = 'rebeccapurple'
colormap = 'BuPu'
```

Importing data and first look

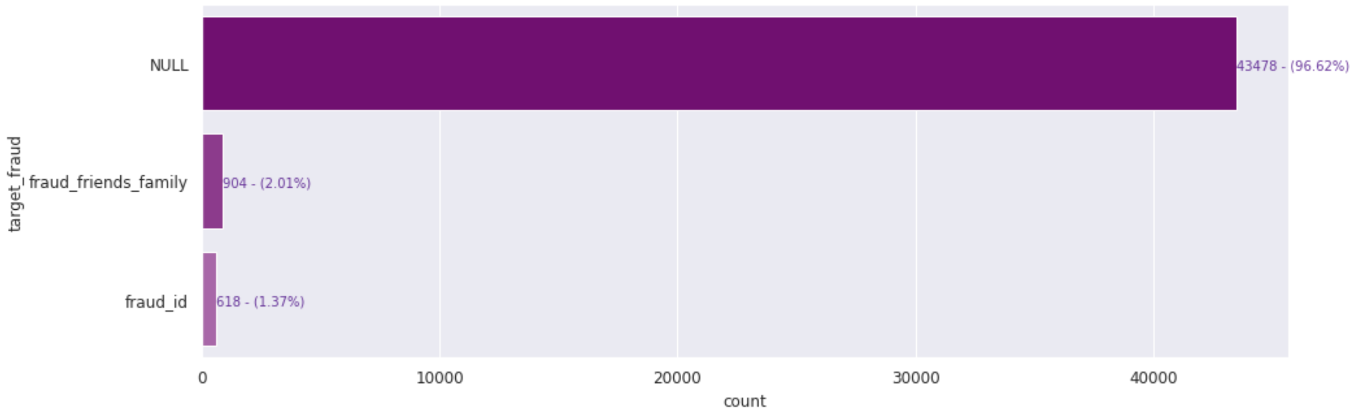
```
new_data_path = '../data/interim/'
```

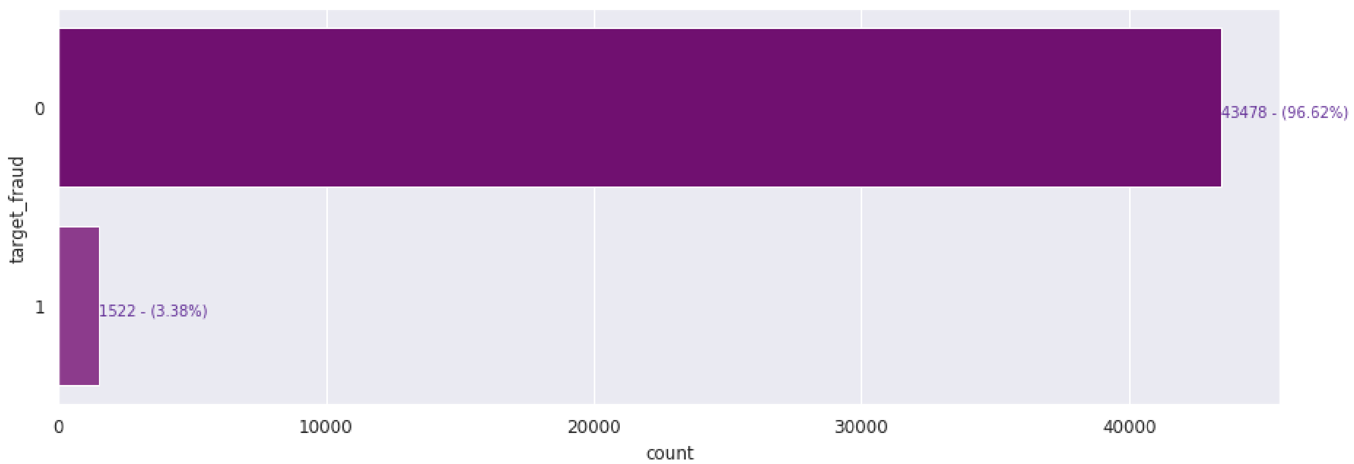
```
df_name = new_data_path + 'acquisition_train.csv'
```

```
df = pd.read_csv(df_name)
df.shape
df.info()
df.describe()
df.head()
```

```
# nulls on target
plot_count(df, 'target_fraud')
df['target_fraud'].fillna('-1', inplace=True)
df['target_fraud'] = df['target_fraud'].apply(lambda x: 0 if x == '-1' else 1)
plot_count(df, 'target_fraud')
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
AxesSubplot(0.125,0.125;0.775x0.755)
```

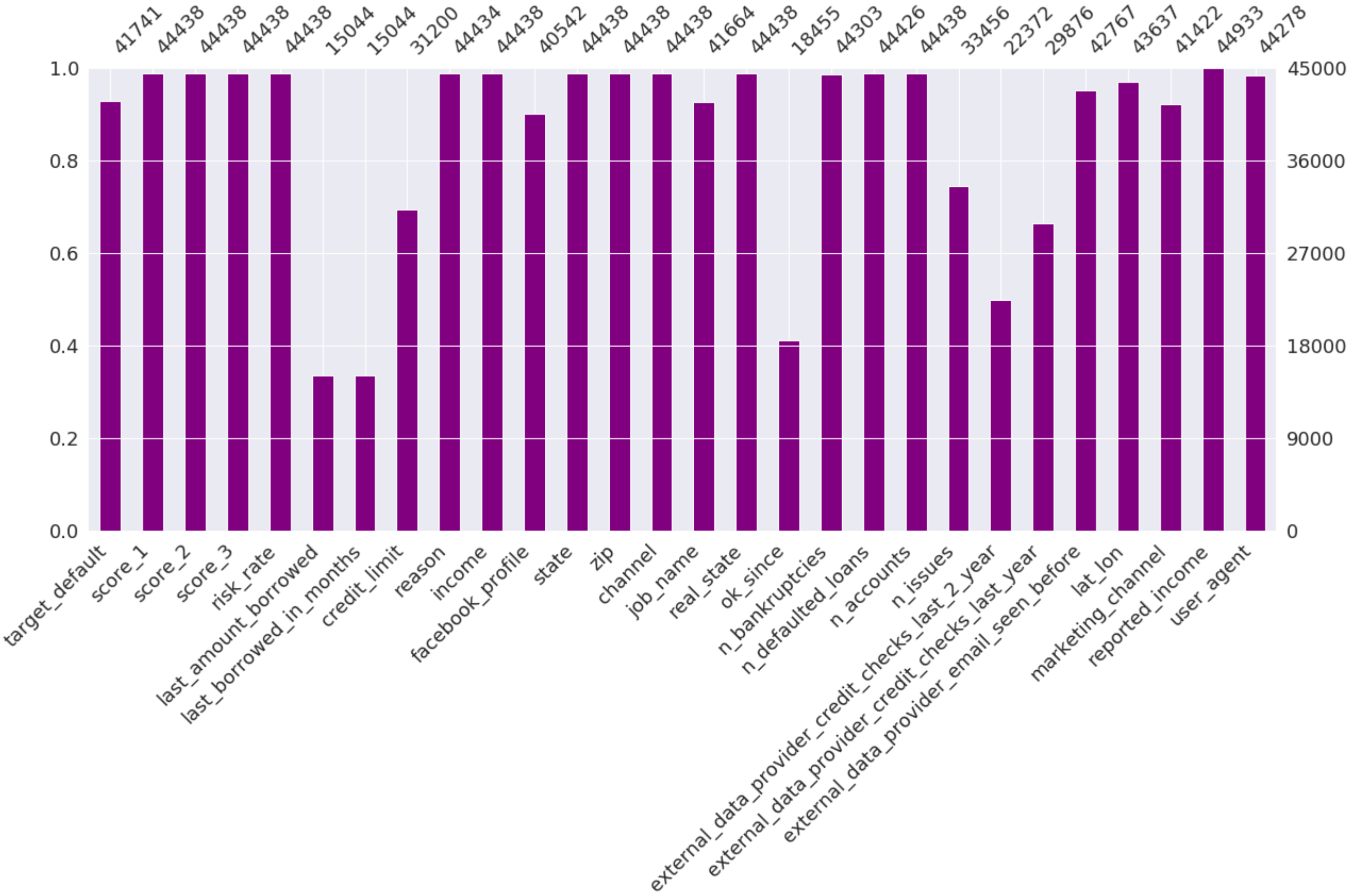


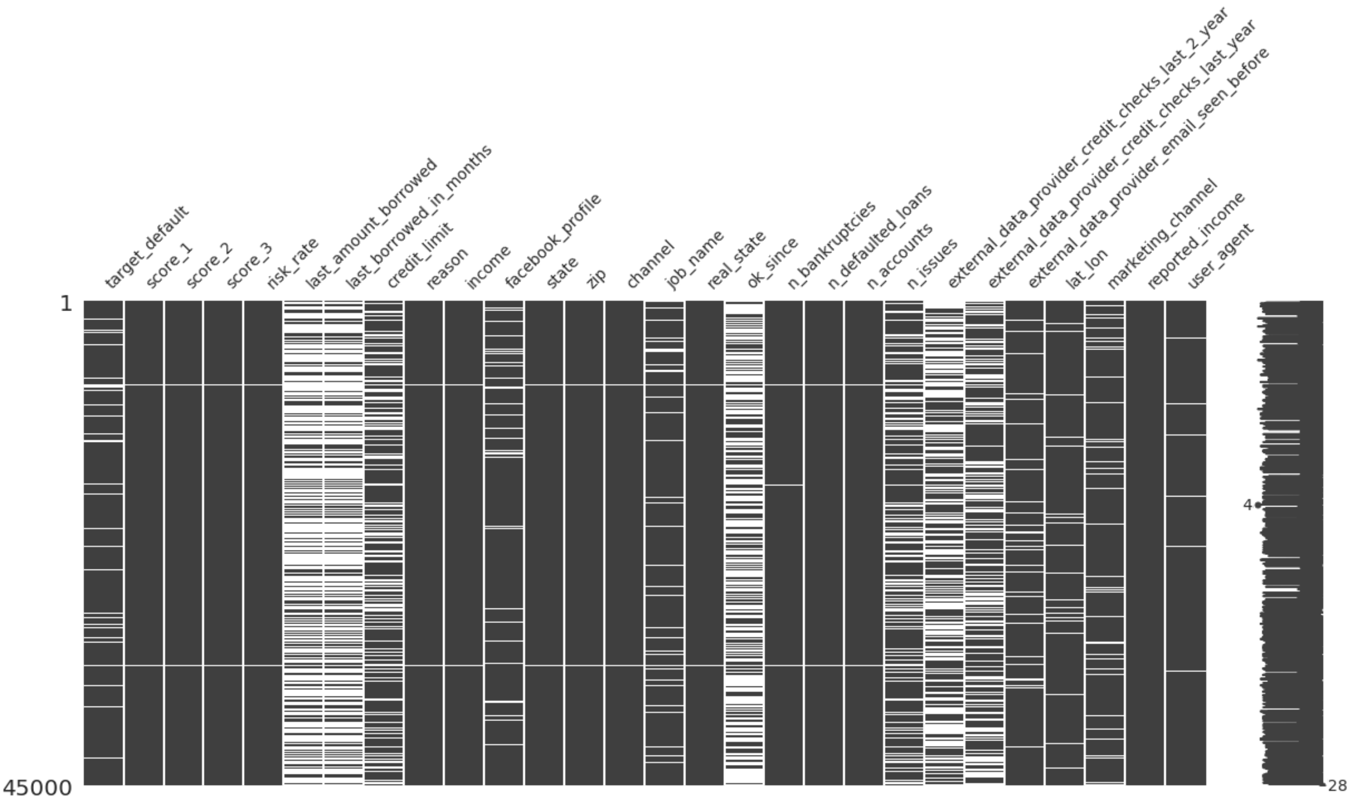


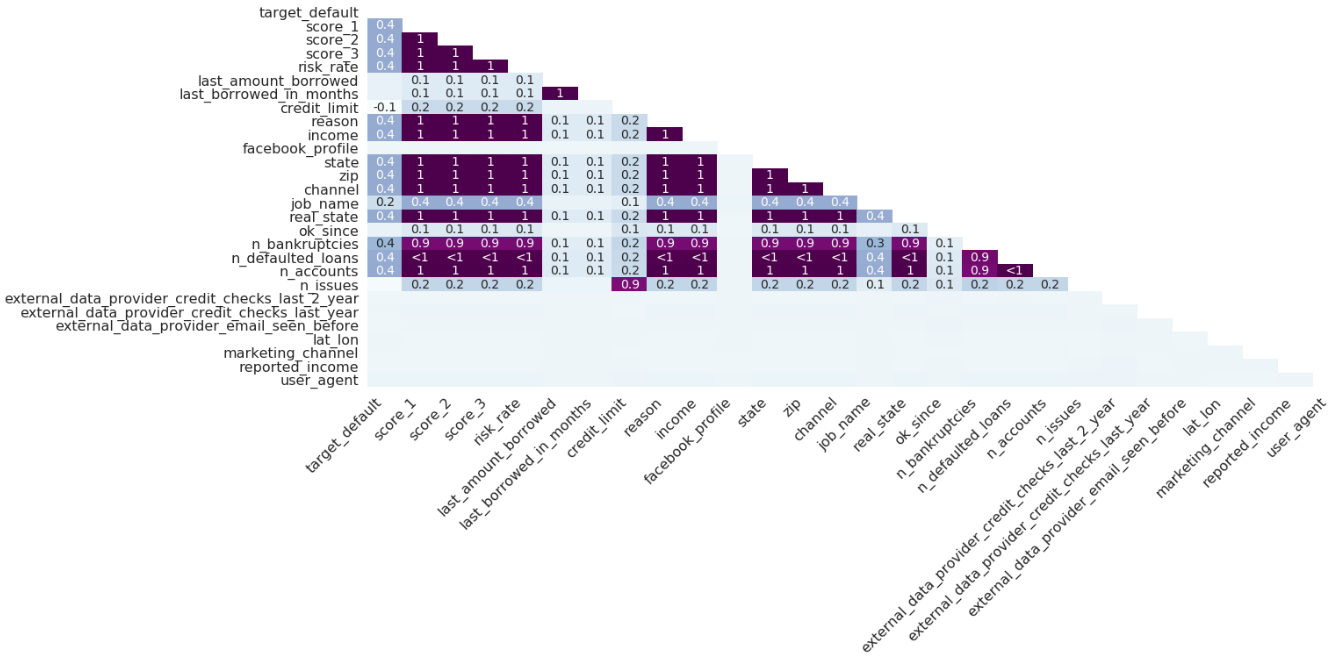
Missing values

```
missing_value_columns = df.columns[df.isnull().any()].tolist()
df_missing = df[missing_value_columns]

msno.bar(df_missing,figsize=(20,8),color=default_color,fontsize=18,labels=True)
msno.matrix(df_missing,figsize=(20,8),fontsize=14)
msno.heatmap(df_missing,figsize=(20,8),cmap=colormap)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e712312e8>
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e723dd390>
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e70386908>
```



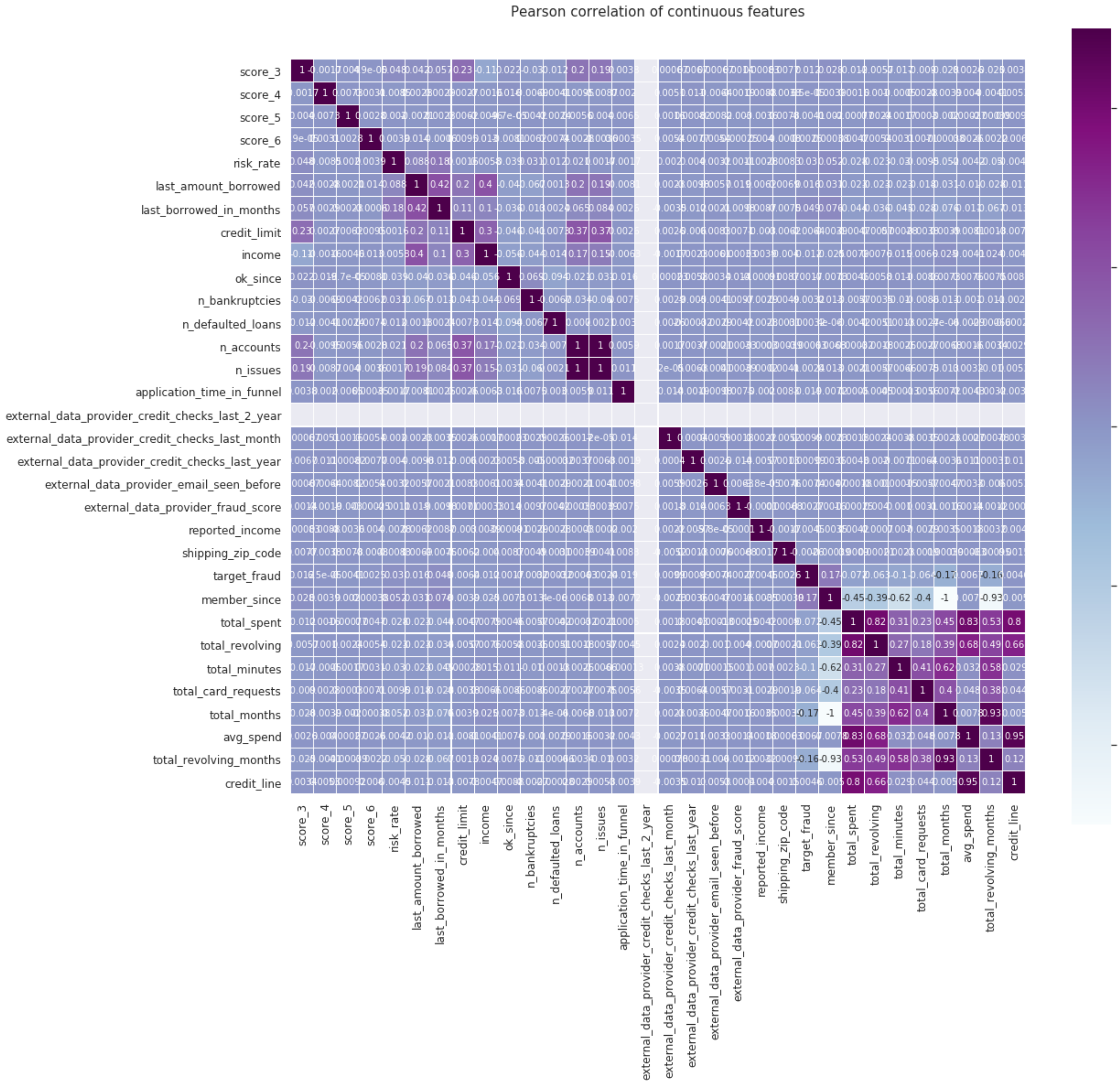




Pearson correlation matrix

```
plt.figure(figsize=(18,16))
plt.title('Pearson correlation of continuous features', y=1.05, size=15)
sns.heatmap(df.corr(),linewidths=0.1,vmax=1.0, square=True, cmap=colormap, linec
olor='white', annot=True)
```

```
<Figure size 1296x1152 with 0 Axes>

Text(0.5,1.05,'Pearson correlation of continuous features')

<matplotlib.axes._subplots.AxesSubplot at 0x7f1e71231630>
```


Pearson correlation of continuous features
score_3
score_4
score_5
score_6
risk_rate
last_amount_borrowed
last_borrowed_in_months
credit_limit
income
ok_since
n_bankruptcies
n_defaulted_loans
n_accounts
n_issues
application_time_in_funnel
external_data_provider_credit_checks_last_2_year
external_data_provider_credit_checks_last_month
external_data_provider_credit_checks_last_year
external_data_provider_email_seen_before
external_data_provider_fraud_score
reported_income
shipping_zip_code
target_fraud
member_since
total_spent
total_revolving
total_minutes
total_card_requests
total_months
avg_spend
total_revolving_months
credit_line
0.8
0.4
0.0
−0.4
−0.8

Drop features

This features do not contribute too much to this model, let's drop them.

```
# unecessary columns
drop_cols = [
            'ids', 'credit_limit', 'channel', 'reason', 'job_name', 'reason'
            'external_data_provider_first_name', 'profile_phone_number',
            'avg_spend', 'target_default', 'facebook_profile', 'profile_tags',
            'last_amount_borrowed', 'last_borrowed_in_months',
            'zip', 'email', 'user_agent', 'n_issues',
            'application_time_applied', 'application_time_in_funnel',
            'external_data_provider_credit_checks_last_2_year',
            'external_data_provider_credit_checks_last_month',
            'external_data_provider_credit_checks_last_year',
            'external_data_provider_first_name',
            'class', 'member_since', 'credit_line',
            'total_spent', 'total_revolving', 'total_minutes',
            'total_card_requests', 'total_months', 'total_revolving_months']

for col in drop_cols:
    if col in df.columns:
        df.drop(col, axis=1, inplace=True)
```

Dealing with Missing values

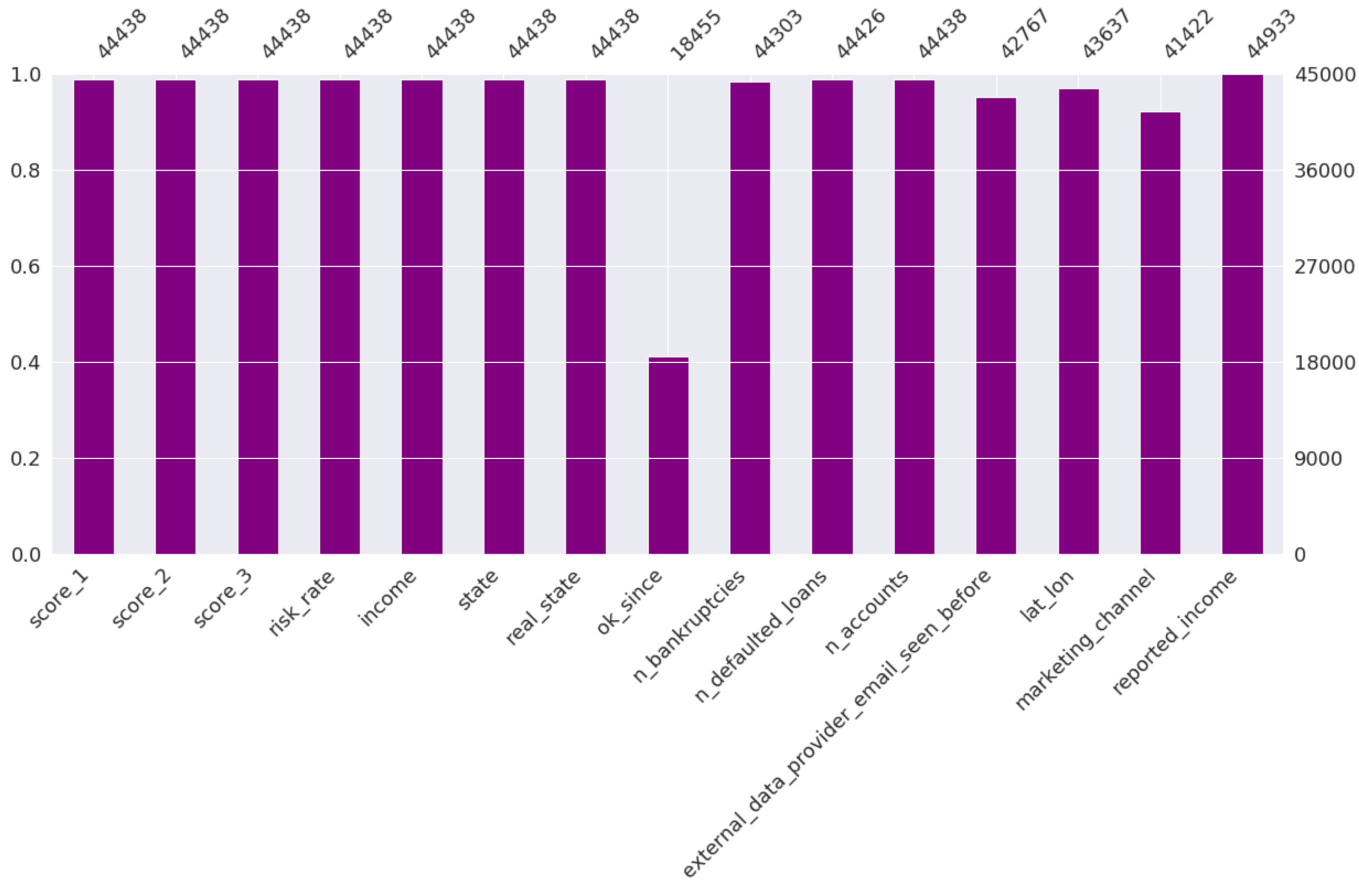
First let's take a look into missing values. Them let's treat each one in the best way possible.

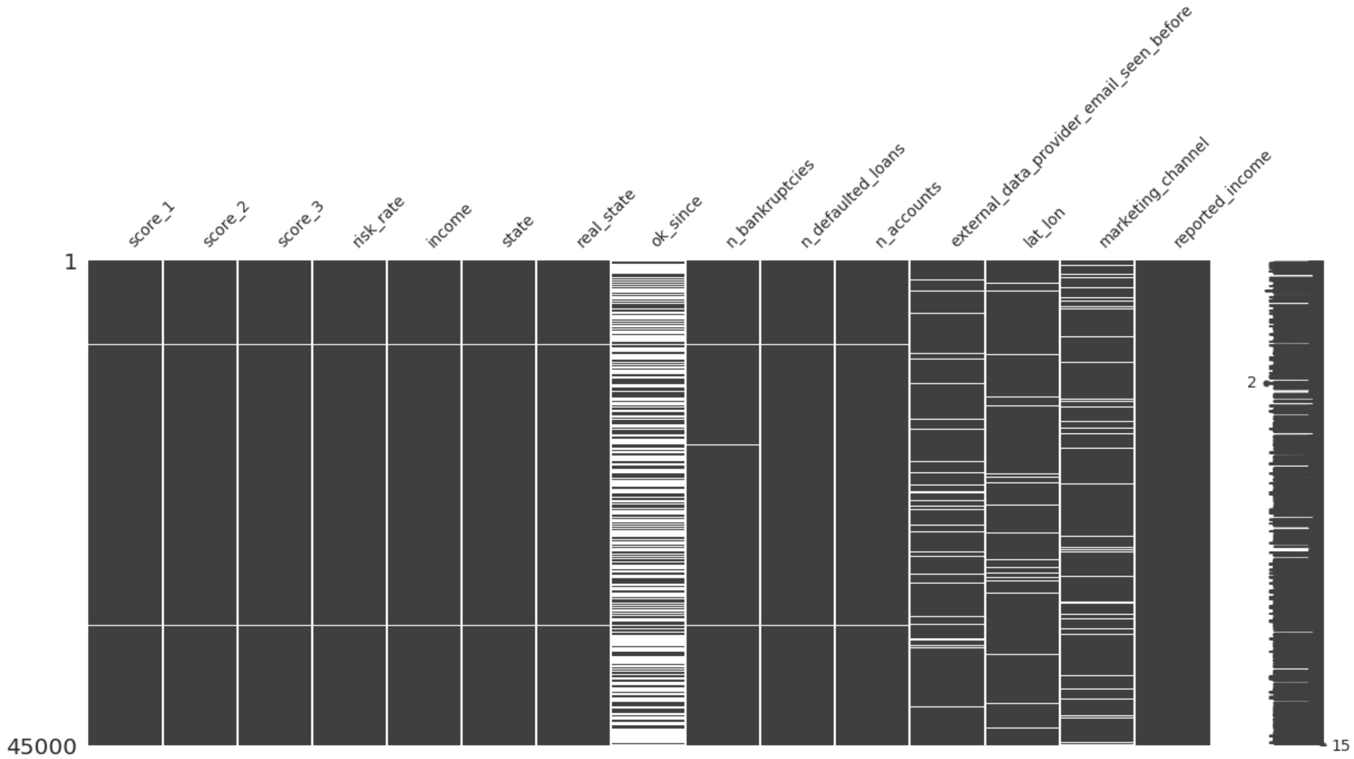
```
missing_value_columns = df.columns[df.isnull().any()].tolist()
df_missing = df[missing_value_columns]

msno.bar(df_missing,figsize=(20,8),color=default_color,fontsize=18,labels=True)
msno.matrix(df_missing,figsize=(20,8),fontsize=14)
msno.heatmap(df_missing,figsize=(20,8),cmap=colormap)
```

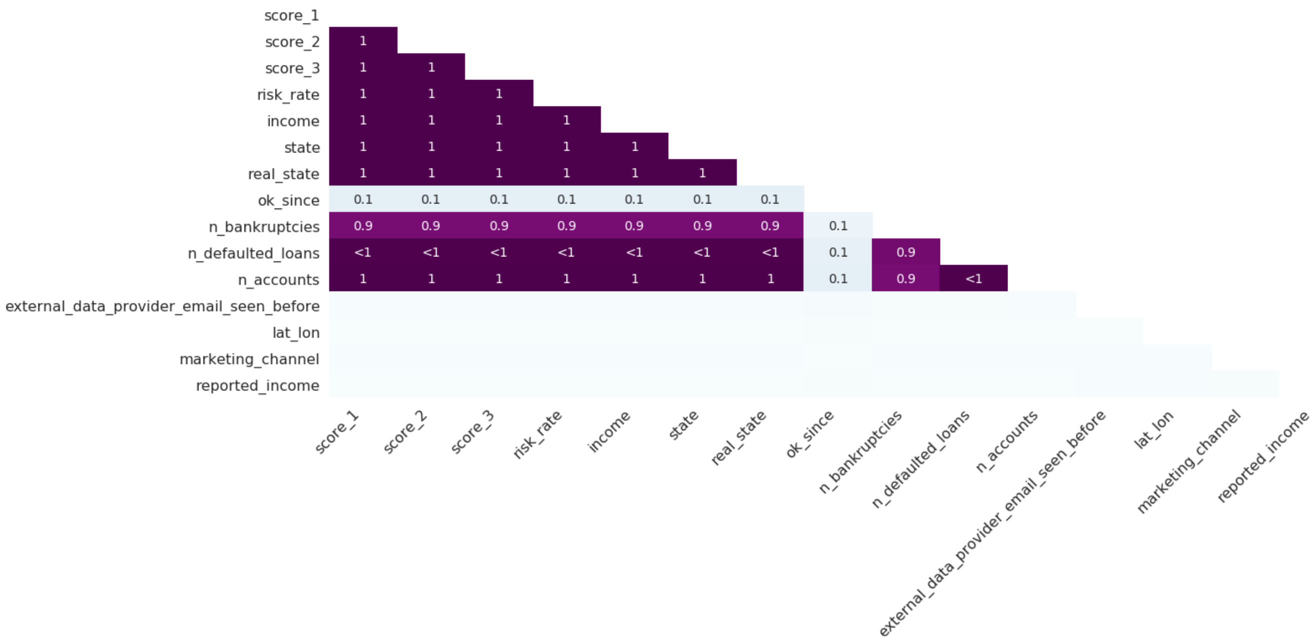
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e7238f400>

<matplotlib.axes._subplots.AxesSubplot at 0x7f1e715310f0>

<matplotlib.axes._subplots.AxesSubplot at 0x7f1e711a33c8>




score_1
score_2
score_3
risk_rate
income
state
real_state
ok_since
n_bankruptcies
n_defaulted_loans
n_accounts
external_data_provider_email_seen_before
lat_lon
marketing_channel
reported_income
1
45000
2
15



Fill nulls

```
# fill nulls
df['ok_since'].fillna((df['ok_since'].mean()), inplace=True)
df['n_bankruptcies'].fillna(-1, inplace=True)
df['n_defaulted_loans'].fillna(-1, inplace=True)
df['external_data_provider_email_seen_before'].fillna((df['external_data_provider_email_seen_before'].mean()), inplace=True)
df['reported_income'].fillna((df['reported_income'].mean()), inplace=True)
df['marketing_channel'].fillna('NA', inplace=True)
df['lat_lon'].fillna('(0,0)', inplace=True)
```

Lat Lon

Let's transform lat_lon into two separate columns

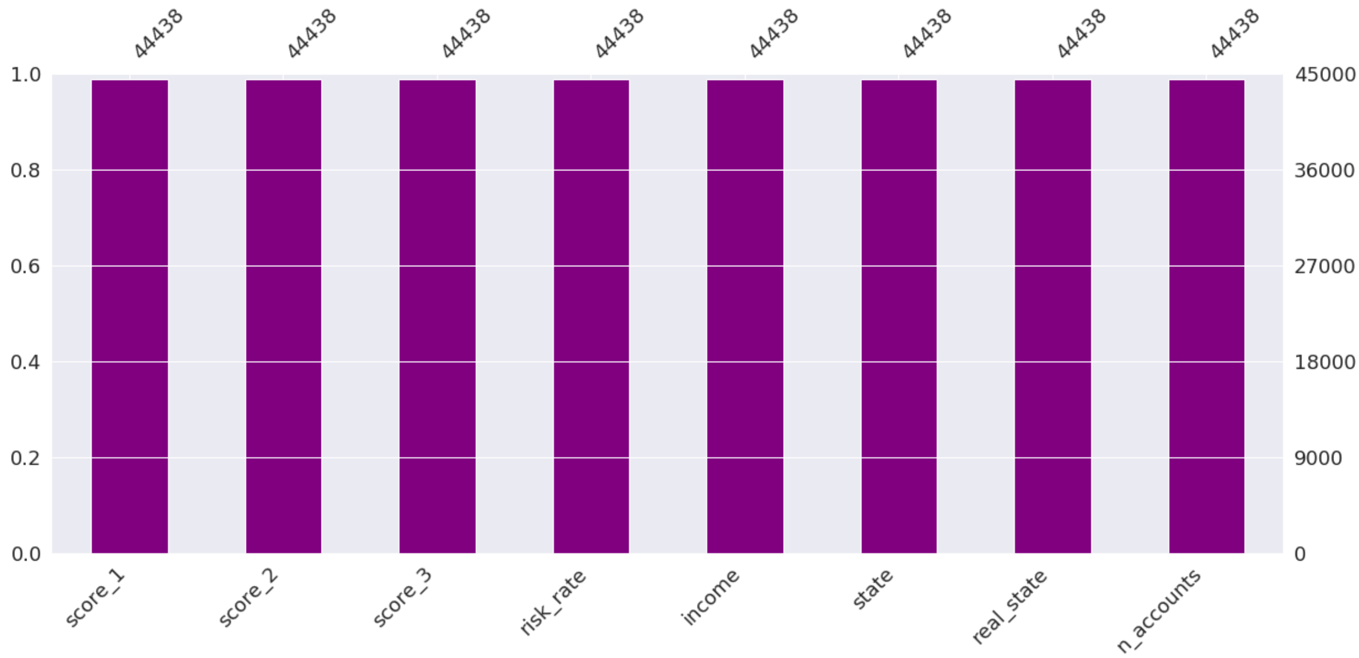
```
# lat lon
df['lat'] = df['lat_lon'].apply(lambda x: ast.literal_eval(x)[0])
df['lon'] = df['lat_lon'].apply(lambda x: ast.literal_eval(x)[1])
df.drop('lat_lon', axis=1, inplace=True)
```

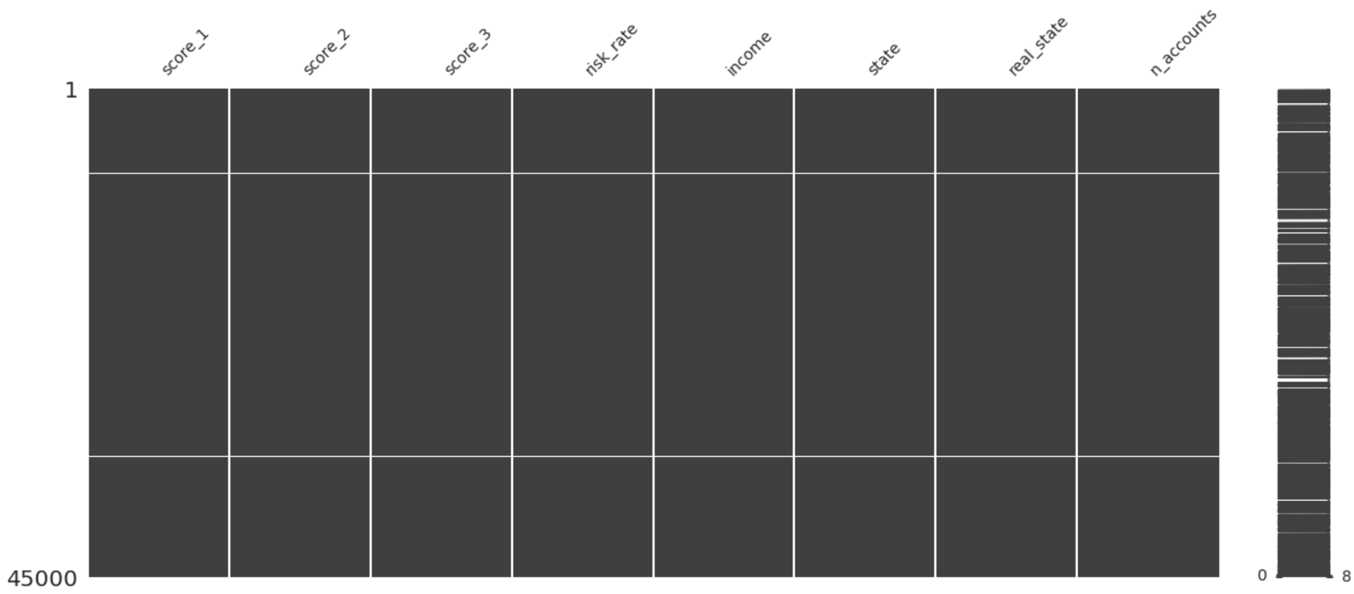
Drop the rest of missing values

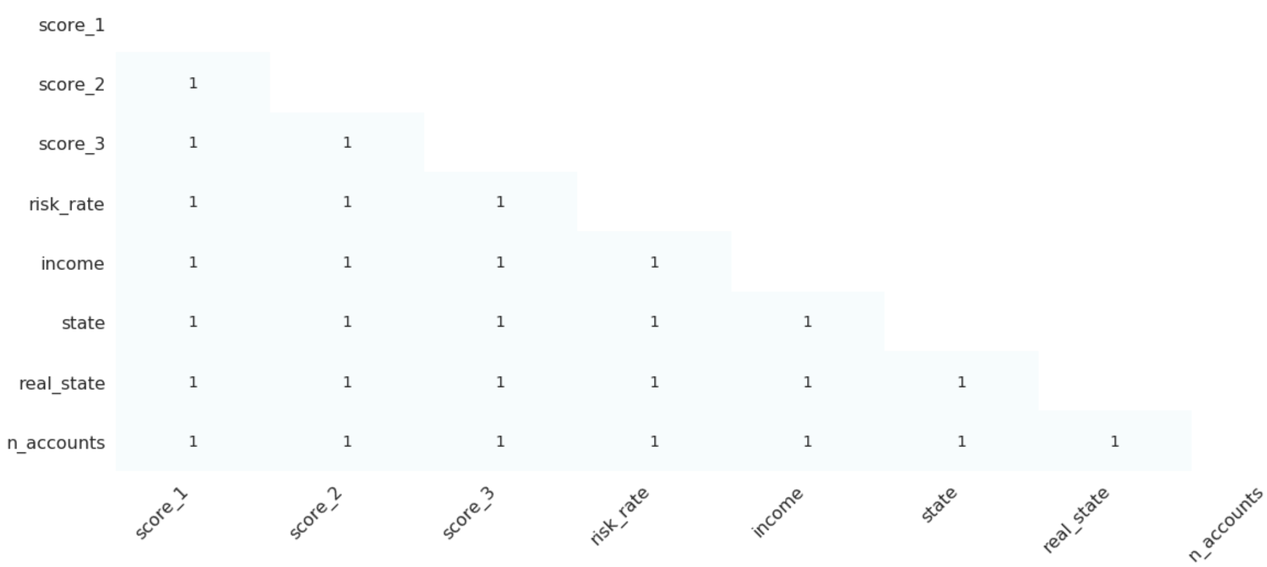
First let's take a look into missing values. Them let's treat each one in the best way possible.

```
missing_value_columns = df.columns[df.isnull().any()].tolist()
if len(missing_value_columns) > 0:
    df_missing = df[missing_value_columns]

    msno.bar(df_missing,figsize=(20,8),color=default_color,fontsize=18,labels=Tr
ue)
    msno.matrix(df_missing,figsize=(20,8),fontsize=14)
    msno.heatmap(df_missing,figsize=(20,8),cmap=colormap)
else:
    print('No Missing values')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e71380b70>
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e7129fe80>
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e70ee5908>
```








```
df.dropna(inplace=True)
df.shape
```

```
(44438, 23)
```

Encoding categorical columns

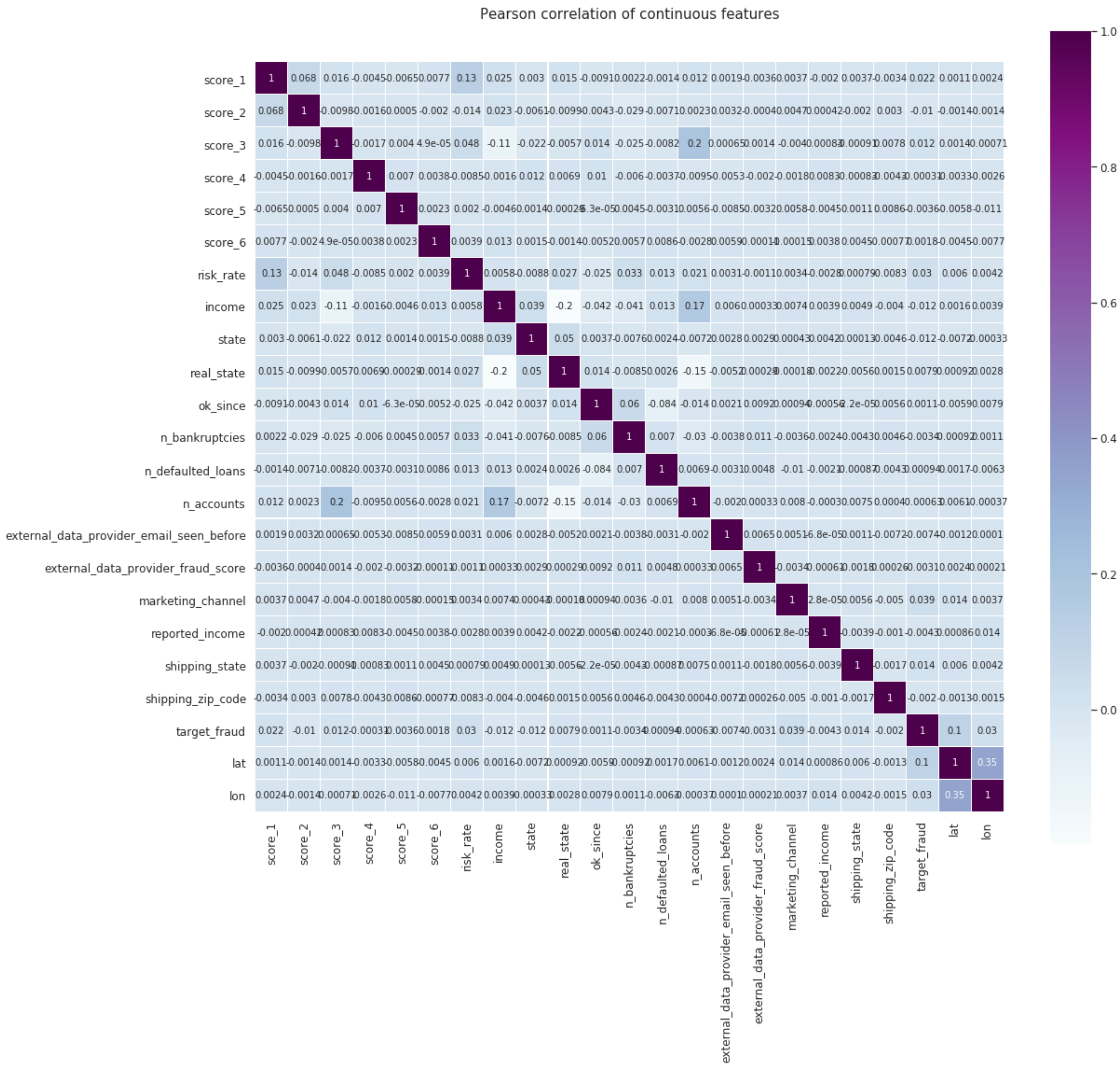
```
encode_columns = [
    'score_1', 'score_2', 'reason', 'state', 'job_name',
    'real_state', 'marketing_channel', 'shipping_state',
    'shipping_zip_code'
]
l_e = LabelEncoder()
for col in encode_columns:
    if col in df.columns:
        df[col] = l_e.fit_transform(df[col])
```

```
plt.figure(figsize=(18,16))
plt.title('Pearson correlation of continuous features', y=1.05, size=15)
sns.heatmap(df.corr(),linewidths=0.1,vmax=1.0, square=True, cmap=colormap, linec
olor='white', annot=True)
```

<Figure size 1296x1152 with 0 Axes>

Text(0.5,1.05,'Pearson correlation of continuous features')

<matplotlib.axes._subplots.AxesSubplot at 0x7f1e723f3da0>



```
one_hot = {c: list(df[c].unique()) for c in df.columns if c not in ['target_defa
ult']}
df = OHE_by_unique(df, one_hot, 7)
```

Creating X and y and trainning models

```
X = df.drop('target_fraud',axis=1)
y = df['target_fraud']
```

XGBoost

```
xgb_params = {}
xgb_params['learning_rate'] = 0.01
xgb_params['n_estimators'] = 750
xgb_params['max_depth'] = 6
xgb_params['colsample_bytree'] = 0.6
xgb_params['min_child_weight'] = 0.6
```

```
xgb_model = XGBClassifier(**xgb_params)
```

```
X_train, y_train, X_val, y_val = cross_val_model(X, y, xgb_model)
```

```
Fit XGBClassifier fold 1
    y train:  Counter({0.0: 28634, 1.0: 991})
    y test:   Counter({0.0: 14317, 1.0: 496})
    cross_score: 0.86319
[[14301    16]
 [  486    10]]
Fit XGBClassifier fold 2
    y train:  Counter({0.0: 28634, 1.0: 991})
    y test:   Counter({0.0: 14317, 1.0: 496})
    cross_score: 0.87425
[[14306    11]
 [  483    13]]
Fit XGBClassifier fold 3
    y train:  Counter({0.0: 28634, 1.0: 992})
    y test:   Counter({0.0: 14317, 1.0: 495})
    cross_score: 0.86164
[[14299    18]
 [  483    12]]
```

Random Forest

```
# RandomForest params
rf_params = {}
rf_params['n_estimators'] = 200
rf_params['max_depth'] = 6
rf_params['min_samples_split'] = 70
rf_params['min_samples_leaf'] = 30
```

```
rf_model = RandomForestClassifier(**rf_params)
```

```
cross_val_model(X, y, rf_model)
```

```
Fit RandomForestClassifier fold 1
    y train:  Counter({0.0: 28634, 1.0: 991})
    y test:   Counter({0.0: 14317, 1.0: 496})
    cross_score: 0.83970
[[14317     0]
 [  496     0]]
Fit RandomForestClassifier fold 2
    y train:  Counter({0.0: 28634, 1.0: 991})
    y test:   Counter({0.0: 14317, 1.0: 496})
    cross_score: 0.85199
[[14317     0]
 [  496     0]]
Fit RandomForestClassifier fold 3
    y train:  Counter({0.0: 28634, 1.0: 992})
    y test:   Counter({0.0: 14317, 1.0: 495})
    cross_score: 0.83281
[[14317     0]
 [  495     0]]
```

```
(array([[  0.,  10., 350., ...,   0.,   0.,   0.],
        [  3.,  16., 370., ...,   0.,   0.,   0.],
        [  0.,  21., 510., ...,   0.,   0.,   0.],
        ...,
        [  6.,  31., 370., ...,   0.,   0.,   0.],
        [  4.,  24., 280., ...,   0.,   0.,   0.],
        [  6.,   5., 240., ...,   0.,   0.,   0.]], dtype=float32),
 array([0., 0., 0., ..., 0., 0., 0.], dtype=float32),
 array([[  3.,   9., 360., ...,   0.,   0.,   0.],
        [  2.,   1., 300., ...,   0.,   0.,   0.],
        [  0.,  21., 250., ...,   0.,   0.,   0.],
        ...,
        [  3.,  15., 210., ...,   0.,   0.,   0.],
        [  0.,   2., 620., ...,   0.,   0.,   0.],
        [  2.,  34., 530., ...,   0.,   0.,   0.]], dtype=float32),
 array([0., 0., 0., ..., 0., 0., 0.], dtype=float32))
```

Stacked models

Why not use both together?

```
log_model = LogisticRegression()
```

```
stack = Ensemble(n_splits=3,
        stacker = log_model,
        base_models = (rf_model, xgb_model))
```

```
y_pred = stack.fit_predict(X, y, X)
```

```
Fit RandomForestClassifier fold 1
Fit RandomForestClassifier fold 2
Fit RandomForestClassifier fold 3
Fit XGBClassifier fold 1
Fit XGBClassifier fold 2
Fit XGBClassifier fold 3
Stacker score: 0.87706
[[42773   178]
 [ 1421    66]]
```

Imbalanced learning

Let's balance this dataset using the technique called SMOTE technique (https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis#SMOTE (https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis#SMOTE))

```
X = df.drop('target_fraud',axis=1)
y = df['target_fraud']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
state=42)
```

```
smt = SMOTE(random_state=42, k_neighbors=1)
X_SMOTE, y_SMOTE = smt.fit_sample(X_train, y_train)
```

```
stack.fit(X_SMOTE, y_SMOTE)
```

```
Fit RandomForestClassifier fold 1
Fit RandomForestClassifier fold 2
Fit RandomForestClassifier fold 3
Fit XGBClassifier fold 1
Fit XGBClassifier fold 2
Fit XGBClassifier fold 3
Stacker score: 0.99343
[[34022   334]
 [ 1165 33191]]
```

```
y_pred = stack.predict(X_test)
confusion_matrix(y_test, np.rint(y_pred))
```

```
array([[8501,   94],
       [ 272,   21]])
```