Loan Default Prediction

Objective

The objective of the model is to build a classification model which can predict a loan default. We will be interested in the prediction and understanding what are the major factors results to a default.

About Data

The dataset is taken from Kaggle. The dataset consists of multiple deteministic factors like borrowe's income, gender, loan pupose etc.

Loading the data.

	ID	year	loan_limit	Gender	approv_in_adv	loan_type	loan_purpose	Credit_Worthiness
0	24890	2019	cf	Sex Not Available	nopre	type1	p1	l1
1	24891	2019	cf	Male	nopre	type2	p1	l1
2	24892	2019	cf	Male	pre	type1	p1	l1
3	24893	2019	cf	Male	nopre	type1	p4	l1
4	24894	2019	cf	Joint	pre	type1	p1	l1

5 rows × 34 columns

```
(148670, 34)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148670 entries, 0 to 148669
Data columns (total 34 columns):
 #   Column                     Non-Null Count   Dtype  
---  ------                     --------------   -----  
 0   ID                         148670 non-null  int64  
 1   year                       148670 non-null  int64  
 2   loan_limit                 145326 non-null  object 
 3   Gender                     148670 non-null  object 
 4   approv_in_adv              147762 non-null  object 
 5   loan_type                  148670 non-null  object 
 6   loan_purpose               148536 non-null  object 
 7   Credit_Worthiness          148670 non-null  object 
 8   open_credit                148670 non-null  object 
 9   business_or_commercial     148670 non-null  object 
 10  loan_amount                148670 non-null  int64  
 11  rate_of_interest           112231 non-null  float64
 12  Interest_rate_spread       112031 non-null  float64
 13  Upfront_charges            109028 non-null  float64
 14  term                       148629 non-null  float64
 15  Neg_ammortization          148549 non-null  object 
 16  interest_only              148670 non-null  object 
 17  lump_sum_payment           148670 non-null  object 
 18  property_value             133572 non-null  float64
 19  construction_type          148670 non-null  object 
 20  occupancy_type             148670 non-null  object 
 21  Secured_by                 148670 non-null  object 
 22  total_units                148670 non-null  object 
 23  income                     139520 non-null  float64
 24  credit_type                148670 non-null  object 
 25  Credit_Score               148670 non-null  int64  
 26  co-applicant_credit_type   148670 non-null  object 
 27  age                        148470 non-null  object 
 28  submission_of_application  148470 non-null  object 
 29  LTV                        133572 non-null  float64
 30  Region                     148670 non-null  object 
 31  Security_Type              148670 non-null  object 
 32  Status                     148670 non-null  int64  
 33  dtir1                      124549 non-null  float64
dtypes: float64(8), int64(5), object(21)
memory usage: 38.6+ MB
```

	ID	year	loan_amount	rate_of_interest	Interest_rate_spread	Upfront_c
count	148670.000000	148670.0	1.486700e+05	112231.000000	112031.000000	109028.0
mean	99224.500000	2019.0	3.311177e+05	4.045476	0.441656	3224.
std	42917.476598	0.0	1.839093e+05	0.561391	0.513043	3251
min	24890.000000	2019.0	1.650000e+04	0.000000	-3.638000	0.0
25%	62057.250000	2019.0	1.965000e+05	3.625000	0.076000	581.4
50%	99224.500000	2019.0	2.965000e+05	3.990000	0.390400	2596.4
75%	136391.750000	2019.0	4.365000e+05	4.375000	0.775400	4812.5
max	173559.000000	2019.0	3.576500e+06	8.000000	3.357000	60000.0

	loan_limit	Gender	approv_in_adv	loan_type	loan_purpose	Credit_Worthiness	open_cr
count	145326	148670	147762	148670	148536	148670	148
unique	2	4	2	3	4	2	
top	cf	Male	nopre	type1	p3	l1	r
freq	135348	42346	124621	113173	55934	142344	148

4 rows × 21 columns

```
ID                           148670
year                              1
loan_limit                        2
Gender                            4
approv_in_adv                     2
loan_type                         3
loan_purpose                      4
Credit_Worthiness                 2
open_credit                       2
business_or_commercial            2
loan_amount                     211
rate_of_interest                131
Interest_rate_spread          22516
Upfront_charges               58271
term                             26
Neg_ammortization                 2
interest_only                     2
lump_sum_payment                  2
property_value                  385
construction_type                 2
occupancy_type                    3
Secured_by                        2
total_units                       4
income                         1001
credit_type                       4
Credit_Score                    401
co-applicant_credit_type          2
age                               7
submission_of_application         2
LTV                            8484
Region                            4
Security_Type                     2
Status                            2
dtir1                            57
dtype: int64
```

```
ID                            0
year                          0
loan_limit                 3344
Gender                        0
approv_in_adv               908
loan_type                     0
loan_purpose                134
Credit_Worthiness             0
open_credit                   0
business_or_commercial        0
loan_amount                   0
rate_of_interest          36439
Interest_rate_spread      36639
Upfront_charges           39642
term                         41
Neg_ammortization           121
interest_only                 0
lump_sum_payment              0
property_value            15098
construction_type             0
occupancy_type                0
Secured_by                    0
total_units                   0
income                     9150
credit_type                   0
Credit_Score                  0
co-applicant_credit_type      0
age                         200
submission_of_application   200
LTV                       15098
Region                        0
Security_Type                 0
Status                        0
dtir1                     24121
dtype: int64
```

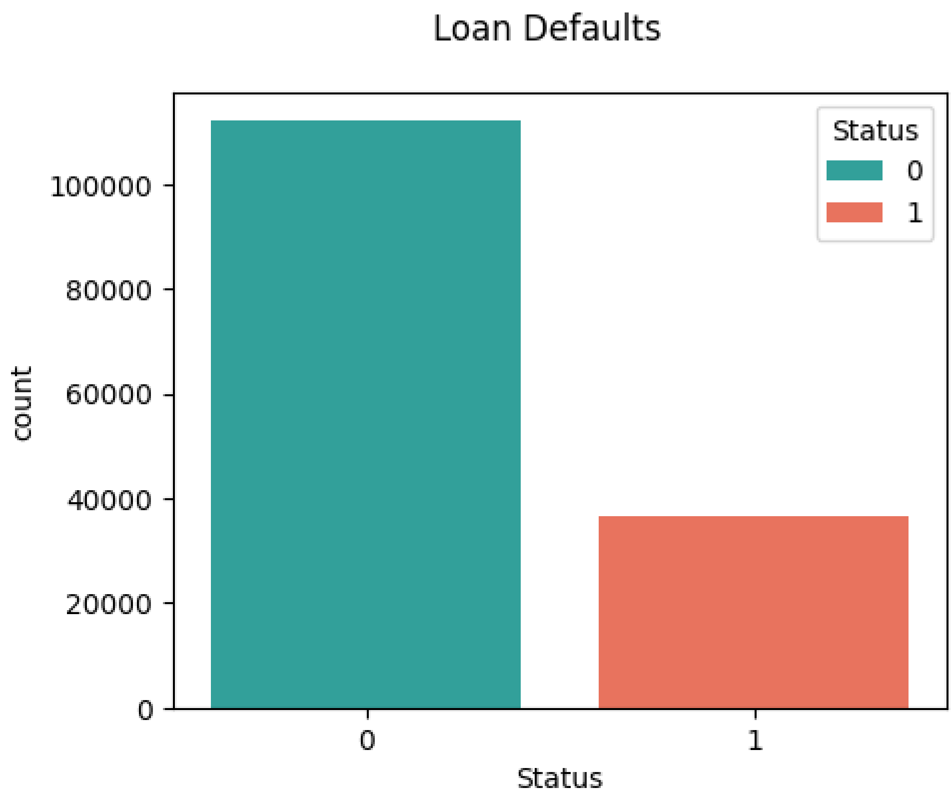
Observations

- Data contains 148670 rows, 34 columns.
- Out of 34, 21 are objects and 13 are numeric.
- 14 columns have missing values.
- We can ignore `ID` and `year` columns. As `ID` is unique to each entry and all the data collected in a single year (2019).
- `Status` will be out target column.
- We will ignore some of features in the cleaning process.

Target Distribution.

Now lets look at how target variable `Status` is distributed.

```
Status
0    112031
1     36639
Name: count, dtype: int64
```




```
The Default percentage is: 24.645%
```

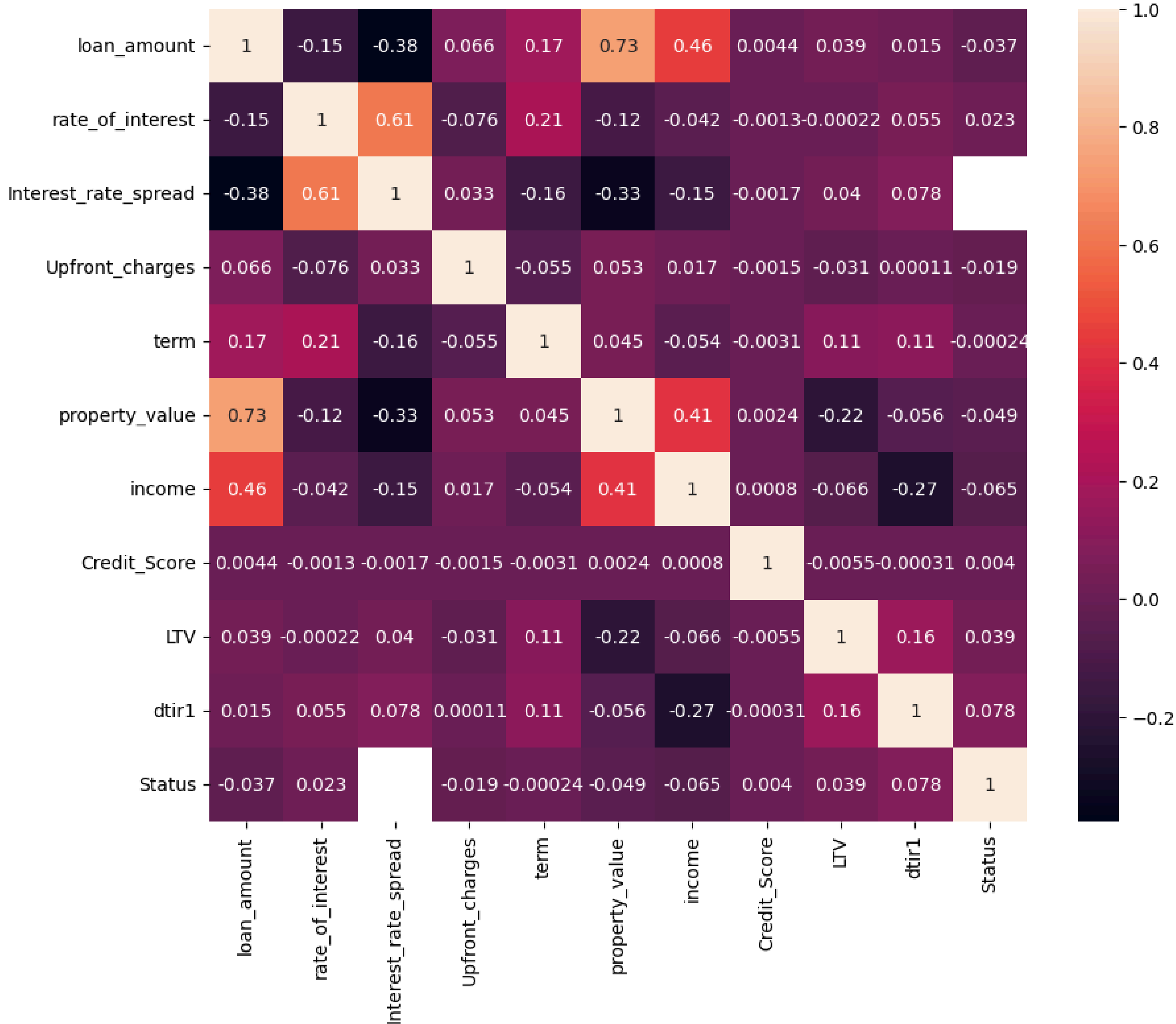
Observations

- There are about 25% defaults. It will be helpful if a loan default is predicted early.
- The target data is imbalanced which might affect the model predictions.

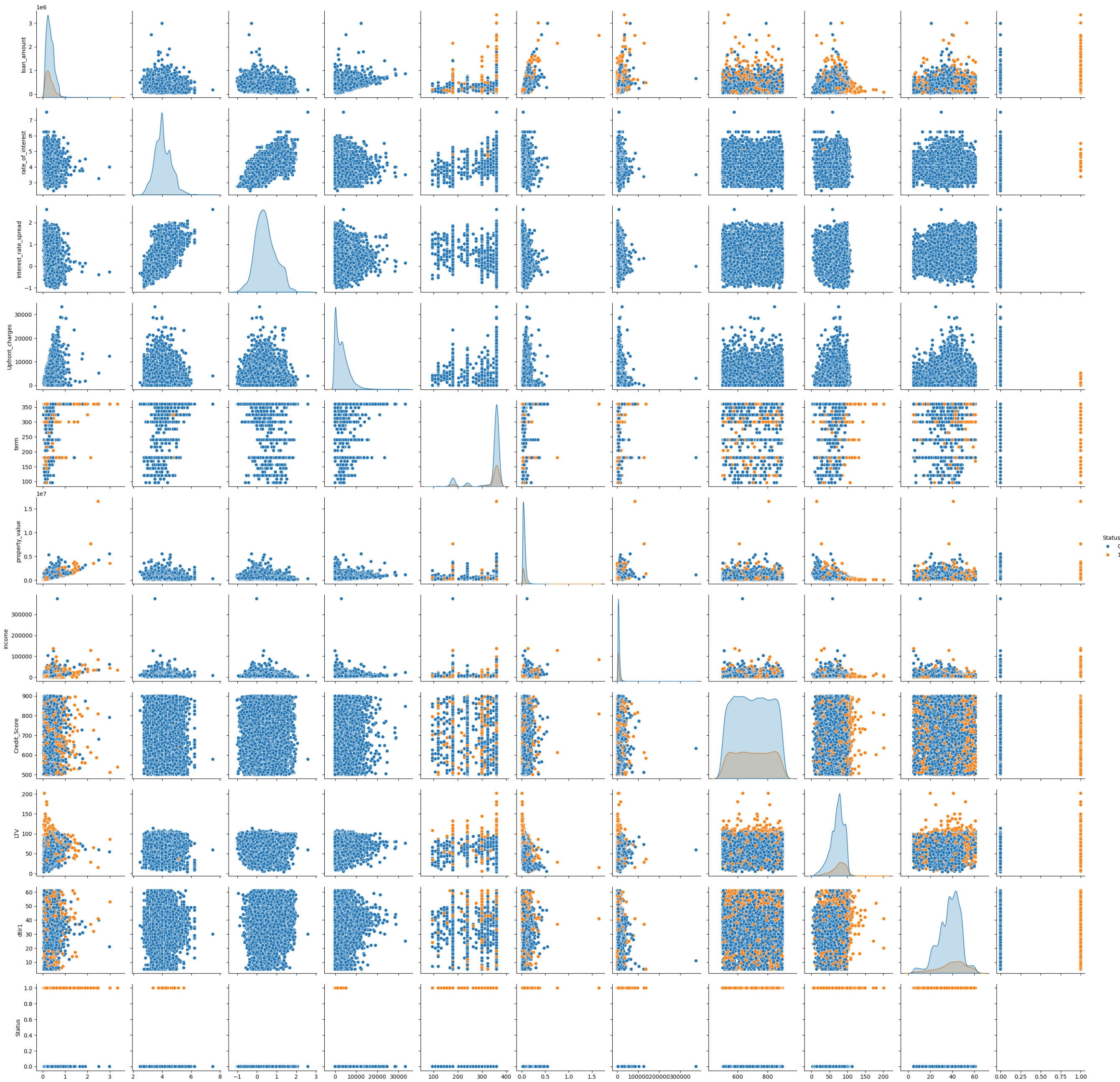
Numerical features

	loan_amount	rate_of_interest	Interest_rate_spread	Upfront_charges	term	property_value
0	116500	NaN	NaN	NaN	360.0	118000.0
1	206500	NaN	NaN	NaN	360.0	NaN
2	406500	4.56	0.2000	595.0	360.0	508000.0
3	456500	4.25	0.6810	NaN	360.0	658000.0
4	696500	4.00	0.3042	0.0	360.0	758000.0

`ID` and `year` are not considered.



loan_amount
rate_of_interest
Interest_rate_spread
Upfront_charges
term
property_value
income
Credit_Score
LTV
dtir1
Status
1 -0.15 -0.38 0.066 0.17 0.73 0.46 0.0044 0.039 0.015 -0.037
-0.15 1 0.61 -0.076 0.21 -0.12 -0.042 -0.0013 -0.00022 0.055 0.023
-0.38 0.61 1 0.033 -0.16 -0.33 -0.15 -0.0017 0.04 0.078
0.066 -0.076 0.033 1 -0.055 0.053 0.017 -0.0015 -0.031 0.00011 -0.019
0.17 0.21 -0.16 -0.055 1 0.045 -0.054 -0.0031 0.11 0.11 -0.00024
0.73 -0.12 -0.33 0.053 0.045 1 0.41 0.0024 -0.22 -0.056 -0.049
0.46 -0.042 -0.15 0.017 -0.054 0.41 1 0.0008 -0.066 -0.27 -0.065
0.0044 -0.0013 -0.0017 -0.0015 -0.0031 0.0024 0.0008 1 -0.0055 -0.00031 0.004
0.039 -0.00022 0.04 -0.031 0.11 -0.22 -0.066 -0.0055 1 0.16 0.039
0.015 0.055 0.078 0.00011 0.11 -0.056 -0.27 -0.00031 0.16 1 0.078
-0.037 0.023 -0.019 -0.00024 -0.049 -0.065 0.004 0.039 0.078 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



```
Status           1.000000
dtir1            0.078083
income           0.065119
property_value   0.048864
LTV              0.038895
loan_amount      0.036825
Name: Status, dtype: float64
```

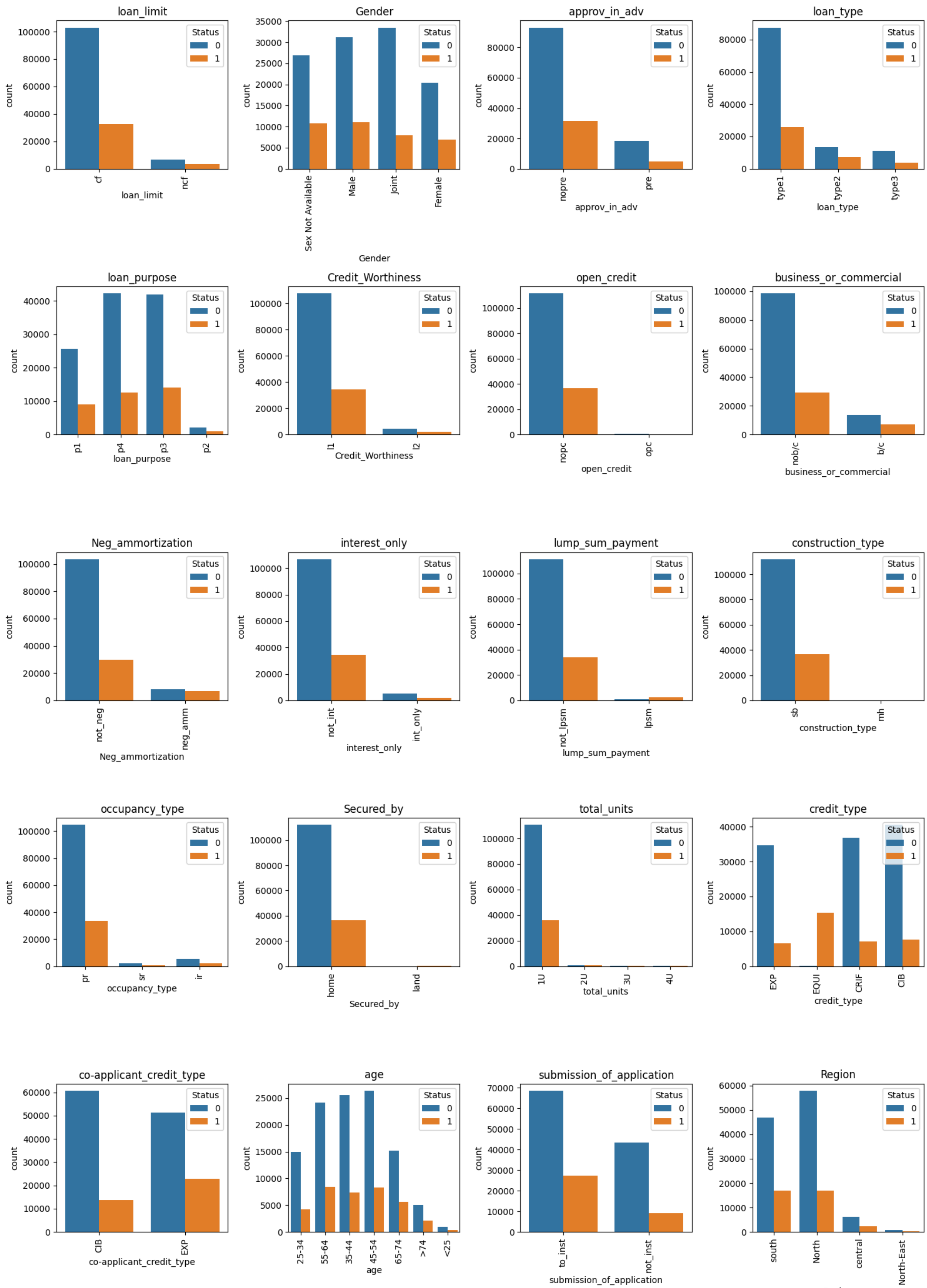
Observations

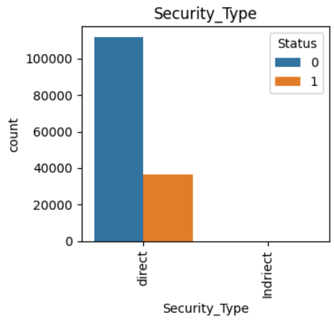
- We see very little correlation of the features with the targets.
- The top five features correlated to target are
  - `dtir1`
  - `income`

    - `property value`
    - `LTV`
    - `loan_amount`
- `rate_of_interest` and `interest_rate_spread` are correlated.
- `loan_amount`, `property_value` and `income` are correlated.
- From pair plot we see a almost linear relations between target and features.
- Here we select only the above numerical features to train the model.

Categorical features

Relationship Between Categorical Variables and Target.


loan_limit
cf
ncf
loan_limit
Gender
Sex Not Available
Male
Joint
Female
Gender
approv_in_adv
nopre
pre
approv_in_adv
loan_type
type1
type2
type3
loan_type
loan_purpose
p1
p4
p3
p2
loan_purpose
Credit_Worthiness
l1
l2
Credit_Worthiness
open_credit
nopc
opc
open_credit
business_or_commercial
nob/c
b/c
business_or_commercial
Neg_ammortization
not_neg
neg_amm
Neg_ammortization
interest_only
not_int
int_only
interest_only
lump_sum_payment
not_lpsm
lpsm
lump_sum_payment
construction_type
sb
mh
construction_type
occupancy_type
pr
sr
ir
occupancy_type
Secured_by
home
land
Secured_by
total_units
1U
2U
3U
4U
total_units
credit_type
EXP
EQUI
CRIF
CIB
credit_type
co-applicant_credit_type
CIB
EXP
co-applicant_credit_type
age
25-34
55-64
35-44
45-54
65-74
>74
<25
age
submission_of_application
to_inst
not_inst
submission_of_application
Region
south
North
central
North-East
Region
count
Status
0
1



- We can ignore the features: `construction_type`, `Security_Type`, `open_credit`, `total_units`, `Secured_by` as they have un even categorical distributuions.
- `submission_of_application`, `age` can also be ignored, since we can argue that they are not much influencing features for default.
- `Gender` is also not included.

Selected features

```
num_features = ['income', 'loan_amount', 'dtir1', 'LTV', 'property_value']
cat_features = ['loan_limit', 'approv_in_adv', 'loan_type', 'loan_purpose',
                'Credit_Worthiness', 'business_or_commercial',
'Neg_ammortization',
                'interest_only', 'lump_sum_payment', 'occupancy_type',
'credit_type',
                'co-applicant_credit_type', 'Region']
```

Handling Missing Values

```
Shape of the data: (148670, 34)

Missing values in Numerical features in percentage:
dtir1             16.224524
LTV               10.155378
property_value    10.155378
income             6.154571
loan_amount        0.000000
dtype: float64

Missing values in Categorical features in percentage:
```

```
loan_limit                  2.249277
approv_in_adv               0.610749
loan_purpose                0.090133
Neg_ammortization           0.081388
loan_type                   0.000000
Credit_Worthiness           0.000000
business_or_commercial      0.000000
interest_only               0.000000
lump_sum_payment            0.000000
occupancy_type              0.000000
credit_type                 0.000000
co-applicant_credit_type    0.000000
Region                      0.000000
dtype: float64
```

- There are about 16% missing data in `dtir1` and about 10% in both `LTV`, `property_value` and 6% in `income`.
- `loan_limit` has about 2.25% missing data and less than 1% data is missing in `approv_in_adv`, `loan_purpose`, `Neg_ammortization`.
- We will impute the numerical data with median values categorical data with mode or most frequent values.
- For Numerical data we stadardize with standard scalar.
- We one-hot encode the categorical data.
- We create pipelines for these.

Data Preparation

```python
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

Importing the above from sklearn library.

Creating two transformers (pipelines) each one for categorical and numerical columns.

```python
cat_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('one_hot', OneHotEncoder(handle_unknown='ignore'))])

num_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())])
```

Combine these into a column transformer

```python
final_transformer = ColumnTransformer(transformers=[
    ('num', num_transformer, num_features),
    ('cat', cat_transformer, cat_features)])
```

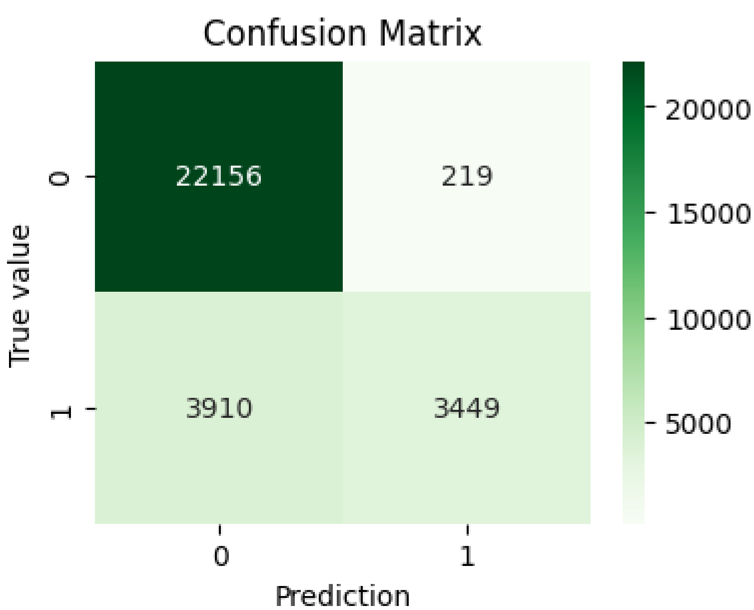
Now, another pipeline with the `final_transformer` and the desired model like `LogisticRegression` can be easily constructed, trained and evaluated.

Logistic Regression

Logistic regression with default parameters.

Training the logistic regression model with default parameters.

```
Accuracy: 0.8611354005515571

Confusion Matrix :
 [[22156   219]
 [ 3910  3449]]

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.99      0.91     22375
           1       0.94      0.47      0.63      7359

    accuracy                           0.86     29734
   macro avg       0.90      0.73      0.77     29734
weighted avg       0.87      0.86      0.84     29734

ROC-AUC Score: 0.8257302903524814
```

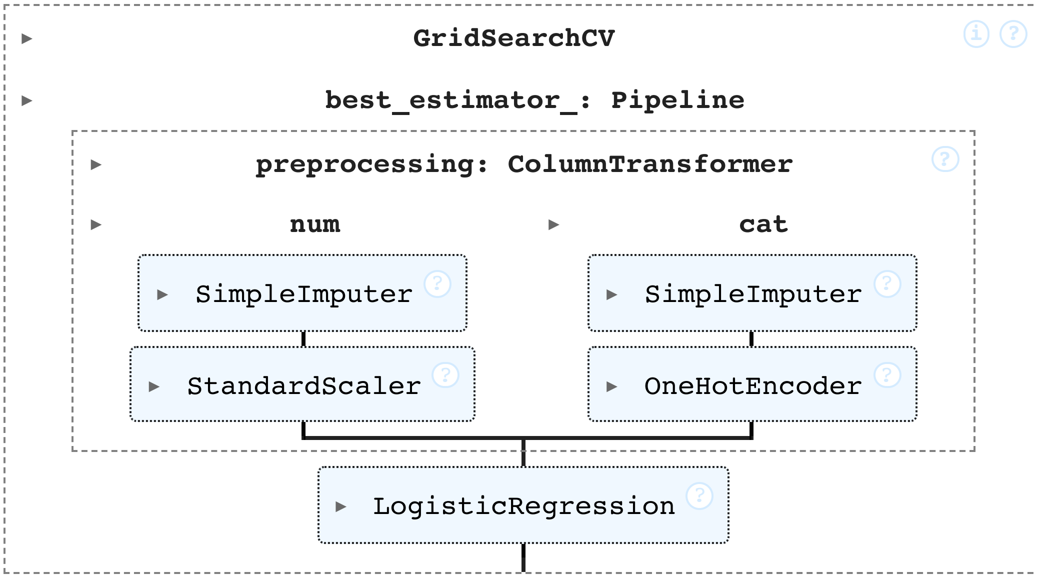


Logistic regression: Finding optimal parameters with Crossvalidation.

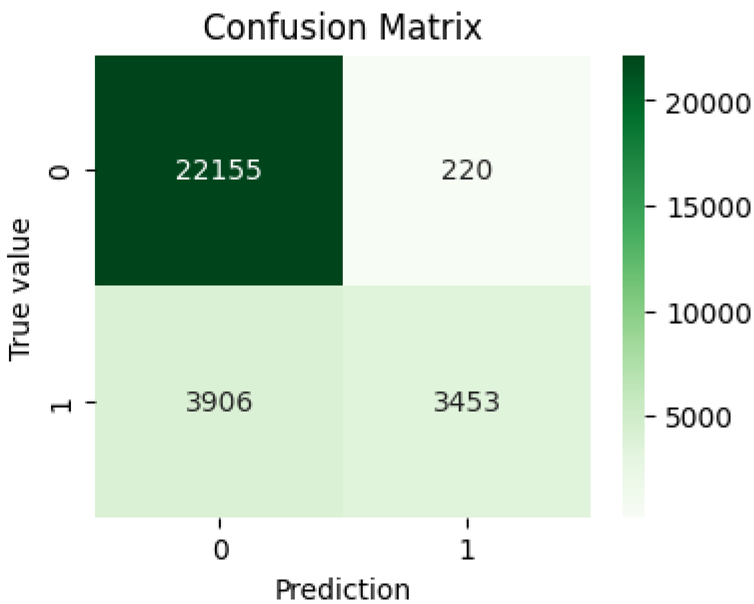
Here, GridSearchCV is used to find optimal values for

- Regularization parameter, C
- penalty
- max_iter

f1_score is used for scoring.




```
Best Parameters:  {'classifier__C': 100, 'classifier__max_iter': 100, 'classifier__pe
nalty': 'l2', 'classifier__solver': 'lbfgs'}
Best f1 Score:  0.6315075253200881

Accuracy:  0.861236295150333

Confusion Matrix:
 [[22155   220]
 [ 3906  3453]]

Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.99      0.91     22375
           1       0.94      0.47      0.63      7359

    accuracy                           0.86     29734
   macro avg       0.90      0.73      0.77     29734
weighted avg       0.87      0.86      0.84     29734


ROC-AUC Score:  0.8259000668812029
```



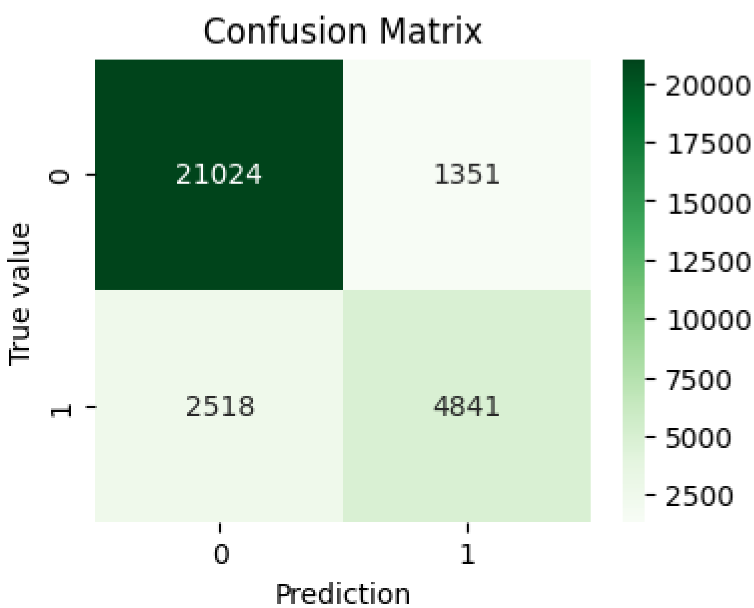
Observations.

- The best estimator with the tuned parameters has
  - Regularization strength 100
  - Maximum iteration 100
  - penalty 'l2'
  - and solver 'lbfgs'
- The best estimator from the GridSerachCV did not perform any differnt way than the model with default parameters.
- This model has high precison and low recall. That is when the model predicts the loan will default, it is about 94% of the times correct. But also from low recall of 47%, the model captures about 47% of the actual defaults.

Random Forests

Now, I train a Random forest model and compare the model performance with the Logistic regression. We also tune the hyperparameters using GridSerachCV.

```
Accuracy: 0.8698795991121275

Confusion Matrix :
 [[21024  1351]
 [ 2518  4841]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.94      0.92     22375
           1       0.78      0.66      0.71      7359

    accuracy                           0.87     29734
   macro avg       0.84      0.80      0.82     29734
weighted avg       0.87      0.87      0.87     29734

ROC-AUC Score: 0.8257302903524814
```



Observations.

- The Random forest model also has the same accuracy of about 87% almost same as the Logistic regression. I optimize the hyperparameter below.

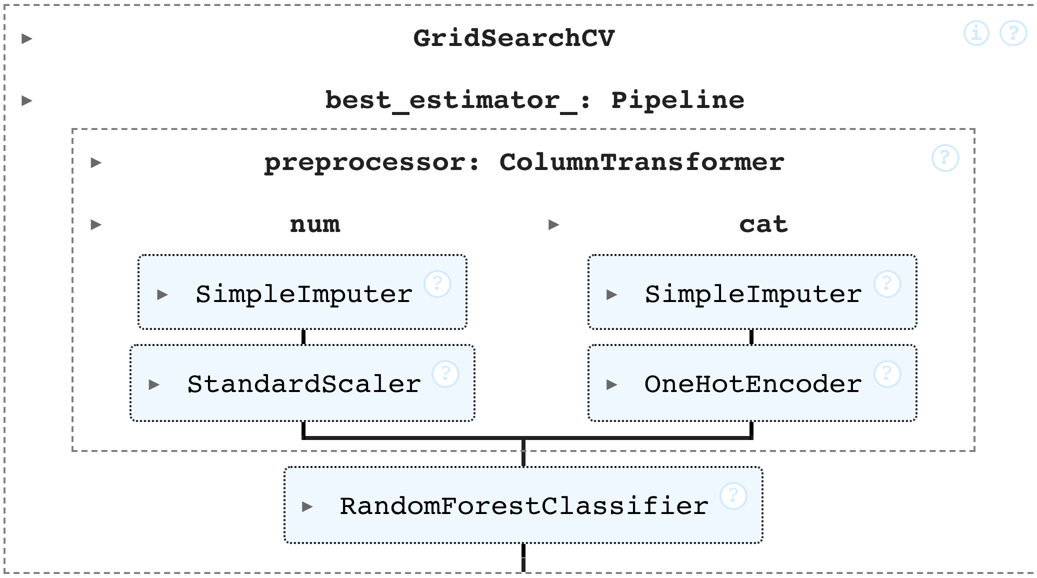
Hyper parameter tuning

The hyperparameters considered for tuning are

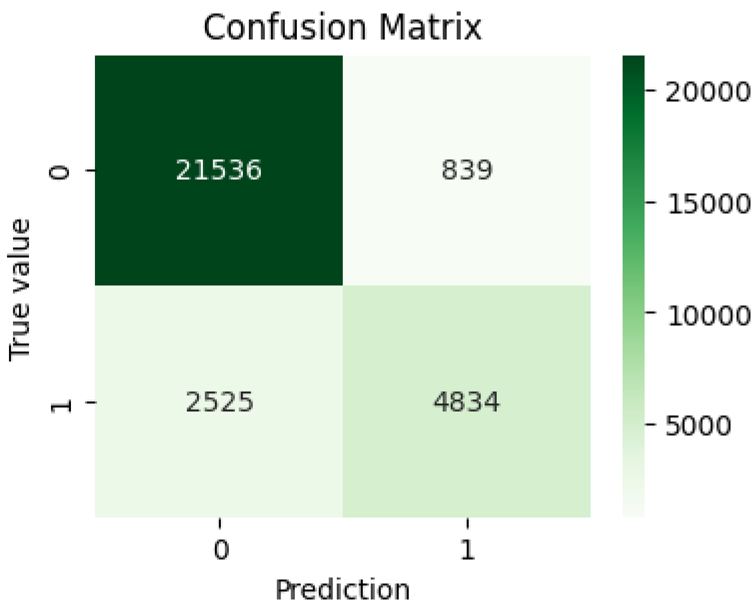
- n_estimators
- max_depth
- min_samples_split
- min_samples_leaf

- max_features

Again, f1-score is used for scoring.




```
Best Parameters:  {'classifier__max_depth': None, 'classifier__max_features': 'sqrt',
'classifier__min_samples_leaf': 1, 'classifier__min_samples_split': 10, 'classifier__
n_estimators': 300}
Best f1 Score:  0.7375981166982257

Accuracy:  0.8868635232393892

Confusion Matrix:
 [[21536   839]
 [ 2525  4834]]

Classification Report:
               precision    recall  f1-score   support

           0       0.90      0.96      0.93     22375
           1       0.85      0.66      0.74      7359

    accuracy                           0.89     29734
   macro avg       0.87      0.81      0.83     29734
weighted avg       0.88      0.89      0.88     29734


ROC-AUC Score:  0.881512098209846
```



Observations

- The best estimator here has
- max_depth: None,
- max_features selected by 'sqrt' method
- min_samples_leaf: 1
- min_samples_split: 10
- n_estimators: 300

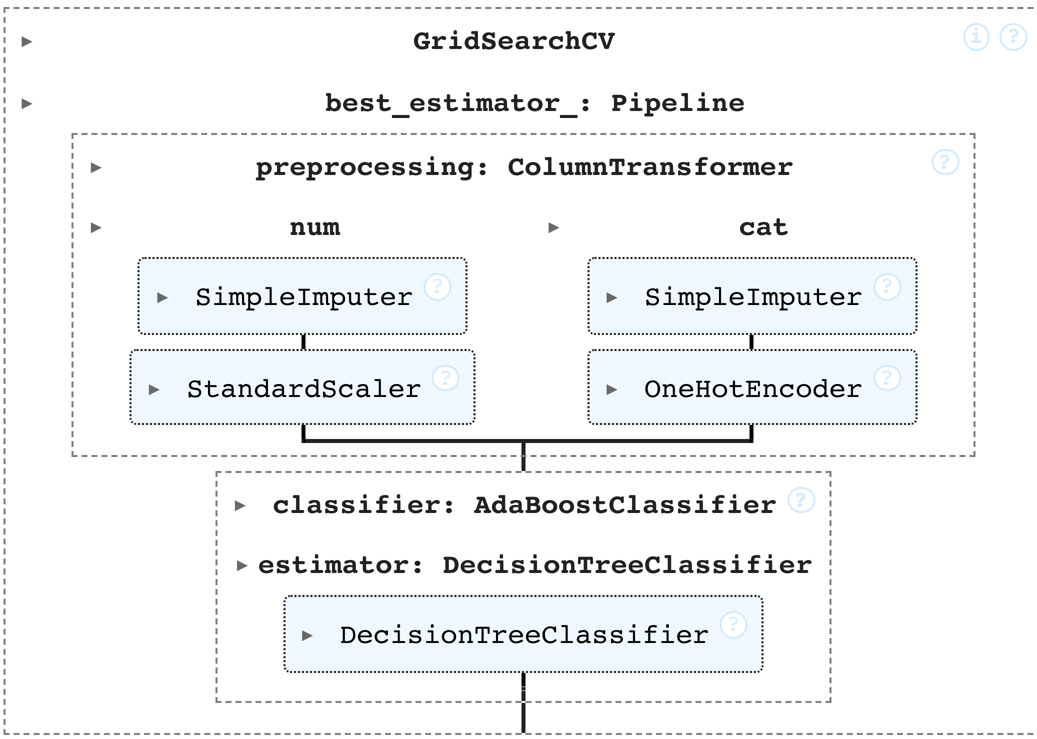
Best f1 Score: 0.7375981166982257

- The Random forest has lower precision than the Logistic regression but has better recall and f1-scores.
- The ROC-AUC score is similar to the Logistic Regression.
- We can choose this model, but for only little improvement we sacrifice the interpretability of the model compared to Logistic regression.

Adaboost Classifier

Here, as the third model, I consider Adaboost with Decision tree as the base estimator.

```
Accuracy: 0.8916055693818524

Confusion Matrix :
 [[21979   396]
 [ 2827  4532]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.98      0.93     22375
           1       0.92      0.62      0.74      7359

    accuracy                           0.89     29734
   macro avg       0.90      0.80      0.83     29734
weighted avg       0.89      0.89      0.88     29734

ROC-AUC Score: 0.8801641709577677
```




```
Best Parameters:  {'classifier__estimator__max_depth': 4, 'classifier__learning_rat
e': 1.0, 'classifier__n_estimators': 19}
Best f1 Score:  0.7366900250011147
```

```
Accuracy:  0.8889823098136813

Confusion Matrix:
 [[21961   414]
 [ 2887  4472]]

Classification Report:
               precision    recall  f1-score   support

           0       0.88      0.98      0.93     22375
           1       0.92      0.61      0.73      7359

    accuracy                           0.89     29734
   macro avg       0.90      0.79      0.83     29734
weighted avg       0.89      0.89      0.88     29734


ROC-AUC Score:  0.8753046844930502
```

The Best Parameters:

- The base estimator decision tree's max_depth': 4,
- learning_rate': 1.0,
- n_estimators': 19 Best f1 Score: 0.7366900250011147

- Adaboost improves over Random forest accuracy and precision.
- This model has the highest f1 and ROC-AUC scores with better predictability.

Which model do I choose?

Here, the problem considers loan defaults. Here model interpretability is very important. Given the scores of all the three models trained here, I would choose the simplest logistic regression model which did almost (may be little less) as good as other two. This is because, it is simple, easy to understand and moreover the model is linear and interpreting the model is very easy.

As a bank, they would like to know what factors affect the most for a given costomer to default.

Further steps to improve model

- I can include more features.
- Since the target has unbalanced classes, we can do oversampling or undersampling.
- We can use a different model like SVM, which may perform better.