HW5 (20 Points): Required Submissions:

1. Submit colab/jupyter notebooks.
2. There are two Questions with different datasets.
3. **You do not need to do EDA again. You can use the EDA from last HW. We are using the same datasets as in the last HW.**
4. Pdf version of the notebooks (HWs will not be graded if pdf version is not provided).
5. **The notebooks and pdf files should have the output.**
6. **Name files as follows : FirstName_file1_hw5, FirstName_file2_h5, FirstName_file3_h5, FirstName_file4_h5**

Question1 (10 Points) : Classification on the 'credit-g' dataset using SVM. OPTIONAL - Try Logistic regression as well.

- **Use RandomSerachCV for this problem**.
- Try poly and rbf kernels in the same pipeline.

Compare KNN (last HW), Logistic Regression/SVM. Basd on your anaysis which algorithm you will recommend.

Download Data:

You can download the dataset using the commands below and see it's description at https://www.openml.org/d/31

Attribute description from https://www.openml.org/d/31

1. Status of existing checking account, in Deutsche Mark.
2. Duration in months
3. Credit history (credits taken, paid back duly, delays, critical accounts)
4. Purpose of the credit (car, television,...)
5. Credit amount
6. Status of savings account/bonds, in Deutsche Mark.
7. Present employment, in number of years.
8. Installment rate in percentage of disposable income
9. Personal status (married, single,...) and sex
10. Other debtors / guarantors
11. Present residence since X years
12. Property (e.g. real estate)
13. Age in years
14. Other installment plans (banks, stores)
15. Housing (rent, own,...)
16. Number of existing credits at this bank
17. Job
18. Number of people being liable to provide maintenance for
19. Telephone (yes,no)
20. Foreign worker (yes,no)

```
import pandas as pd
from scipy.io import arff
from pathlib import Path
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from feature_engine.transformation import YeoJohnsonTransformer
from sklearn.preprocessing import MaxAbsScaler
```

```
A,b = fetch_openml("credit-g", version=1, as_frame=True, return_X_y=True)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The default value of `parser` will c
  warn(
```

```
base = Path("/content/drive/MyDrive/Applied_ML/Class_4/Assignment")
```

```
custom_function_folder = base/"Custom_function"
```

```
sys.path
```

```
['/content',
 '/env/python',
 '/usr/lib/python310.zip',
 '/usr/lib/python3.10',
 '/usr/lib/python3.10/lib-dynload',
 '',
 '/usr/local/lib/python3.10/dist-packages',
 '/usr/lib/python3/dist-packages',
 '/usr/local/lib/python3.10/dist-packages/IPython/extensions',
 '/root/.ipython',
 '/content/drive/MyDrive/Applied_ML/Class_4/Assignment/Custom_functions',
 '/content/drive/MyDrive/Applied_ML/Class_4/Assignment/Custom_function']
```

```
sys.path.append(str(custom_function_folder))
```

```
from eda_plots import diagnostic_plots, plot_target_by_category
```

```
A.head()
```

	checking_status	duration	credit_history	purpose	credit_amount	savings_status	employment	installment_commitmer
0	<0	6.0	critical/other existing credit	radio/tv	1169.0	no known savings	>=7	4
1	0<=X<200	48.0	existing paid	radio/tv	5951.0	<100	1<=X<4	2
2	no checking	12.0	critical/other existing credit	education	2096.0	<100	4<=X<7	2
3	<0	42.0	existing paid	furniture/equipment	7882.0	<100	4<=X<7	2
4	<0	24.0	delayed previously	new car	4870.0	<100	1<=X<4	3

```
A.columns
```

```
Index(['checking_status', 'duration', 'credit_history', 'purpose',
       'credit_amount', 'savings_status', 'employment',
       'installment_commitment', 'personal_status', 'other_parties',
       'residence_since', 'property_magnitude', 'age', 'other_payment_plans',
       'housing', 'existing_credits', 'job', 'num_dependents', 'own_telephone',
       'foreign_worker'],
      dtype='object')
```

```
A.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   checking_status         1000 non-null   category
 1   duration                1000 non-null   float64
 2   credit_history          1000 non-null   category
 3   purpose                 1000 non-null   category
 4   credit_amount           1000 non-null   float64
 5   savings_status          1000 non-null   category
 6   employment              1000 non-null   category
 7   installment_commitment  1000 non-null   float64
 8   personal_status         1000 non-null   category
 9   other_parties           1000 non-null   category
 10  residence_since         1000 non-null   float64
 11  property_magnitude      1000 non-null   category
 12  age                     1000 non-null   float64
 13  other_payment_plans     1000 non-null   category
 14  housing                 1000 non-null   category
 15  existing_credits        1000 non-null   float64
 16  job                     1000 non-null   category
 17  num_dependents          1000 non-null   float64
 18  own_telephone           1000 non-null   category
```

```
 19  foreign_worker          1000 non-null   category
dtypes: category(13), float64(7)
memory usage: 69.9 KB
```

```
A.isnull().any()
```

```
checking_status           False
duration                  False
credit_history            False
purpose                   False
credit_amount             False
savings_status            False
employment                False
installment_commitment    False
personal_status           False
other_parties             False
residence_since           False
property_magnitude        False
age                       False
other_payment_plans       False
housing                   False
existing_credits          False
job                       False
num_dependents            False
own_telephone             False
foreign_worker            False
dtype: bool
```

```
A.nunique()
```

```
checking_status             4
duration                   33
credit_history              5
purpose                    10
credit_amount             921
savings_status              5
employment                  5
installment_commitment      4
personal_status             4
other_parties               3
residence_since             4
property_magnitude          4
age                        53
other_payment_plans         3
housing                     3
existing_credits            4
job                         4
num_dependents              2
own_telephone               2
foreign_worker              2
dtype: int64
```

```
A.describe().T
```

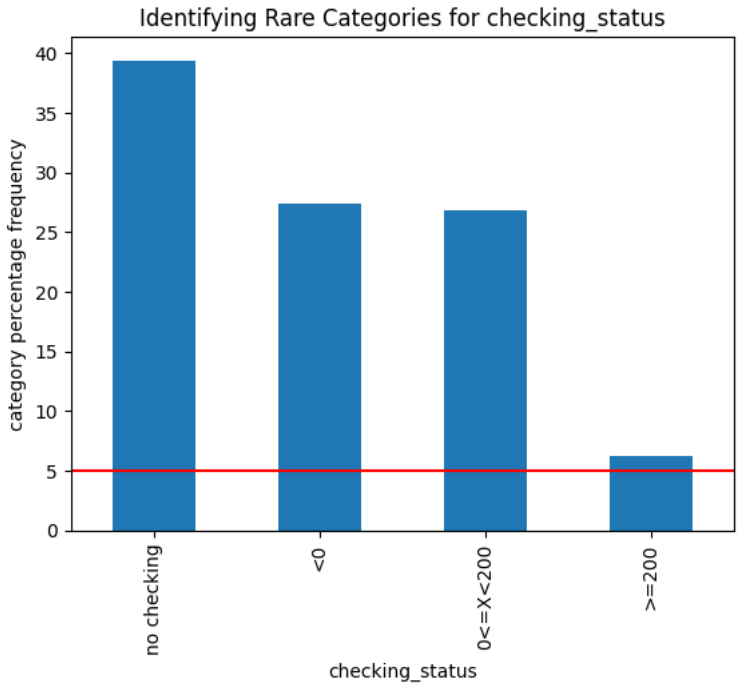
	count	mean	std	min	25%	50%	75%	max
duration	1000.0	20.903	12.058814	4.0	12.0	18.0	24.00	72.0
credit_amount	1000.0	3271.258	2822.736876	250.0	1365.5	2319.5	3972.25	18424.0
installment_commitment	1000.0	2.973	1.118715	1.0	2.0	3.0	4.00	4.0
residence_since	1000.0	2.845	1.103718	1.0	2.0	3.0	4.00	4.0
age	1000.0	35.546	11.375469	19.0	27.0	33.0	42.00	75.0
existing_credits	1000.0	1.407	0.577654	1.0	1.0	1.0	2.00	4.0
num_dependents	1000.0	1.155	0.362086	1.0	1.0	1.0	1.00	2.0

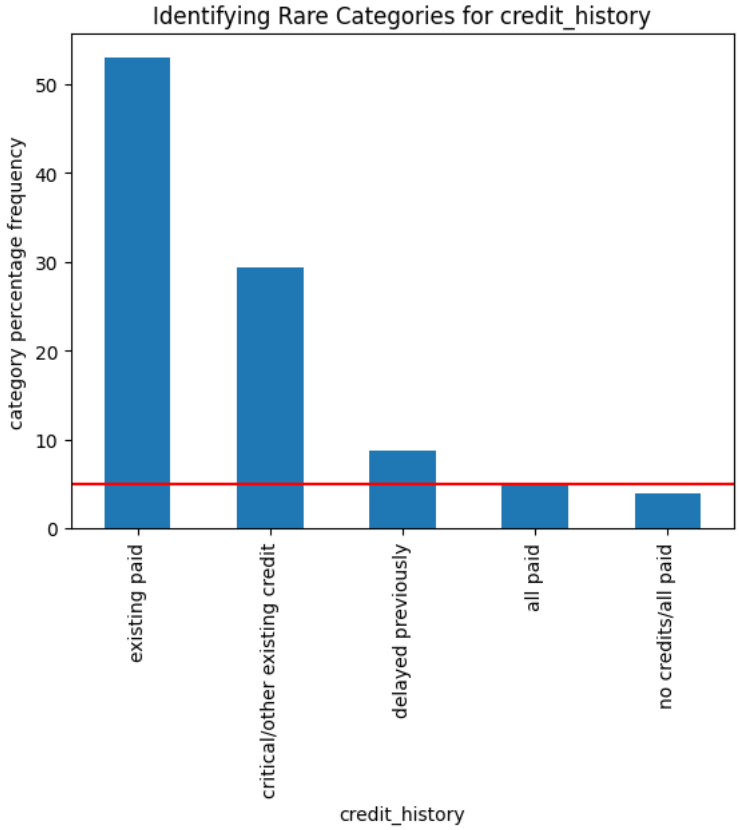
```
categorical_1 = [var for var in A.columns if A[var].dtype == "category"]
discrete_1 = [var for var in A.columns if A[var].dtype != "category" and (len(A[var].unique())< 20)]
continous_1 = [ var for var in A.columns if A[var].dtype != 'category'
              and var not in discrete]
```

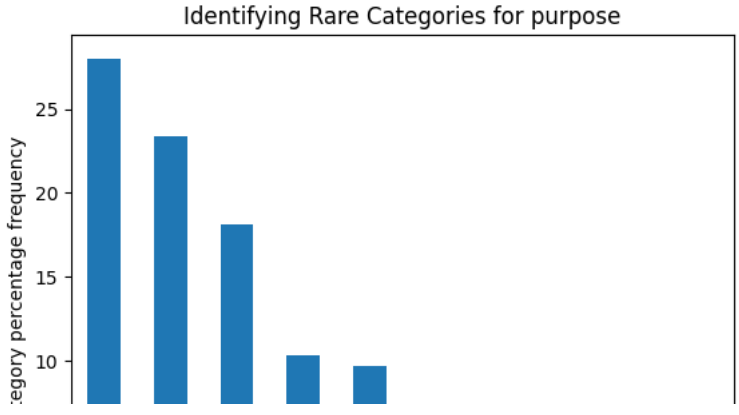
```
continous_1
```

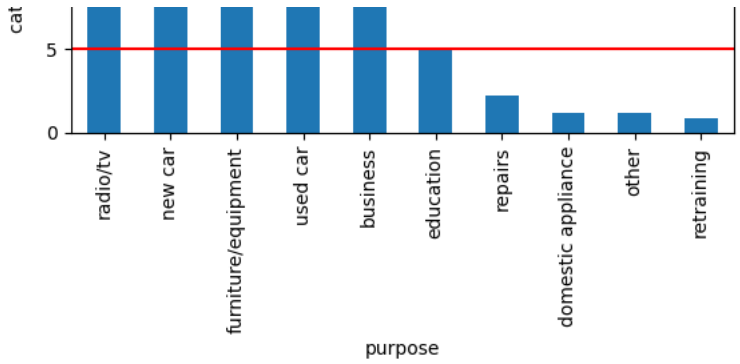
```
['duration', 'credit_amount', 'age']
```

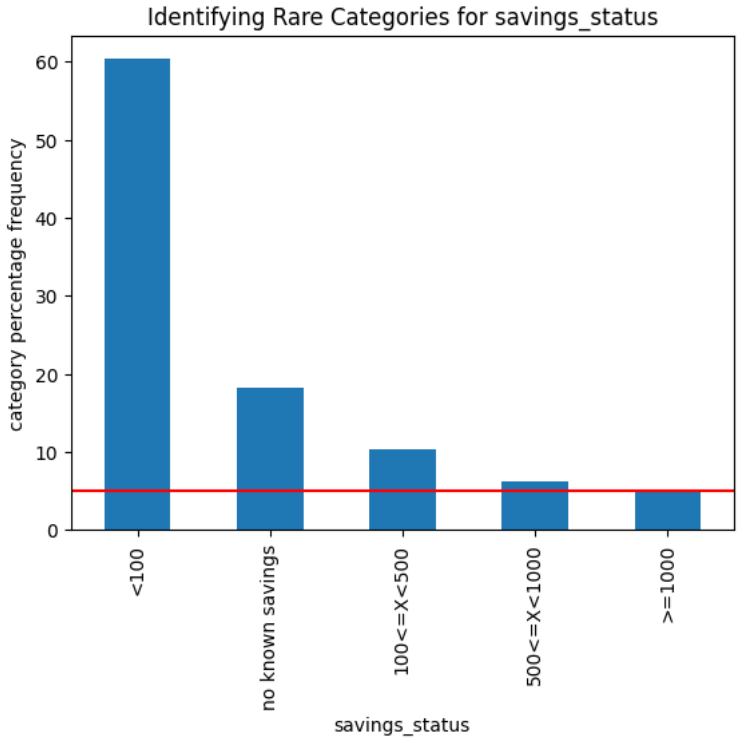
```
def check_rare(df,var):
  cat_freq = 100 * df[var].value_counts(normalize = True)
  fig = cat_freq.sort_values(ascending=False).plot.bar()
  fig.axhline(y=5, color='red')
  fig.set_ylabel('category percentage frequency')
  fig.set_xlabel(var)
  fig.set_title(f'Identifying Rare Categories for {var}')
  plt.show()

for var in categorical_1:
  rare(A,var)
```

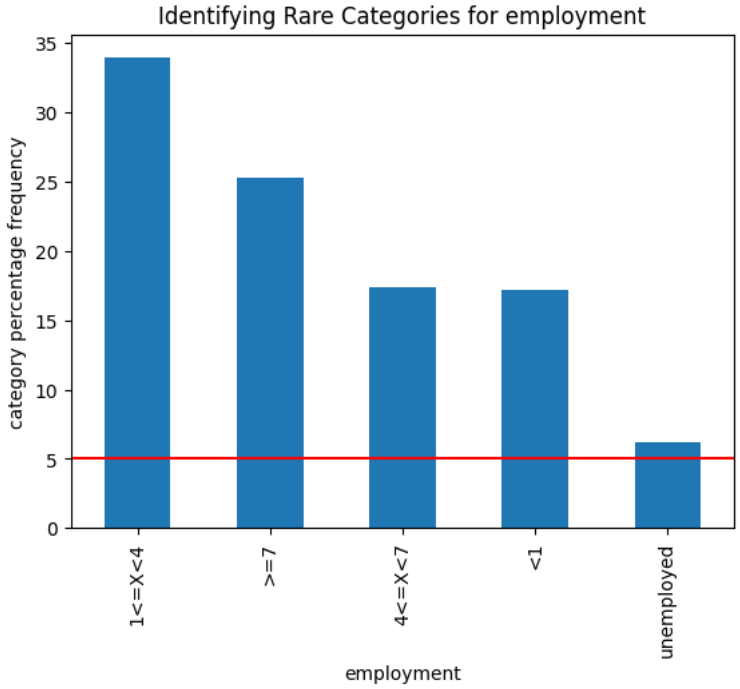

Identifying Rare Categories for checking_status
category percentage frequency
40
35
30
25
20
15
10
5
0
no checking
<0
0<=X<200
>=200
checking_status

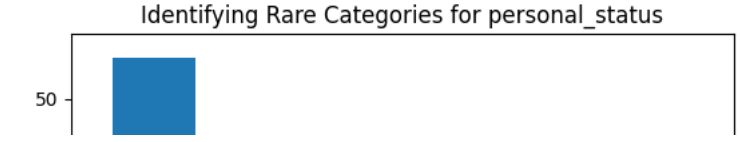

Identifying Rare Categories for credit_history
category percentage frequency
50
40
30
20
10
0
existing paid
critical/other existing credit
delayed previously
all paid
no credits/all paid
credit_history

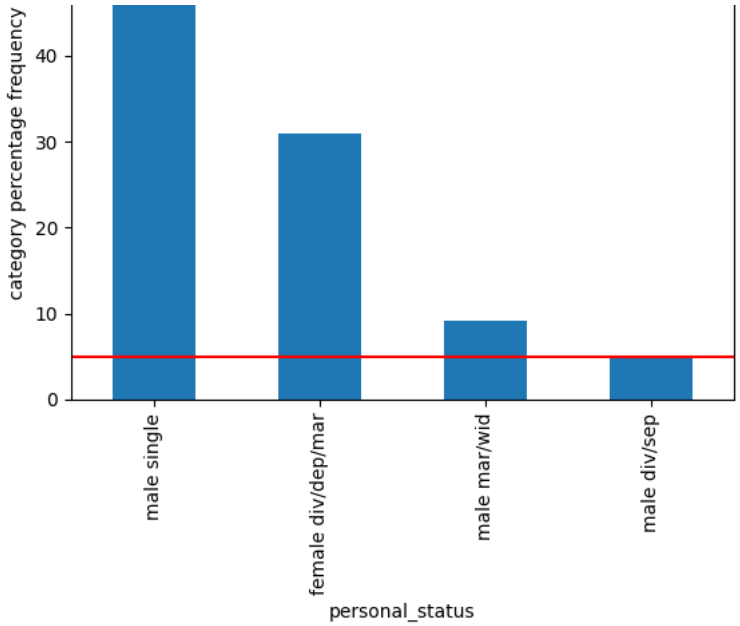

Identifying Rare Categories for purpose
25
20
15
10

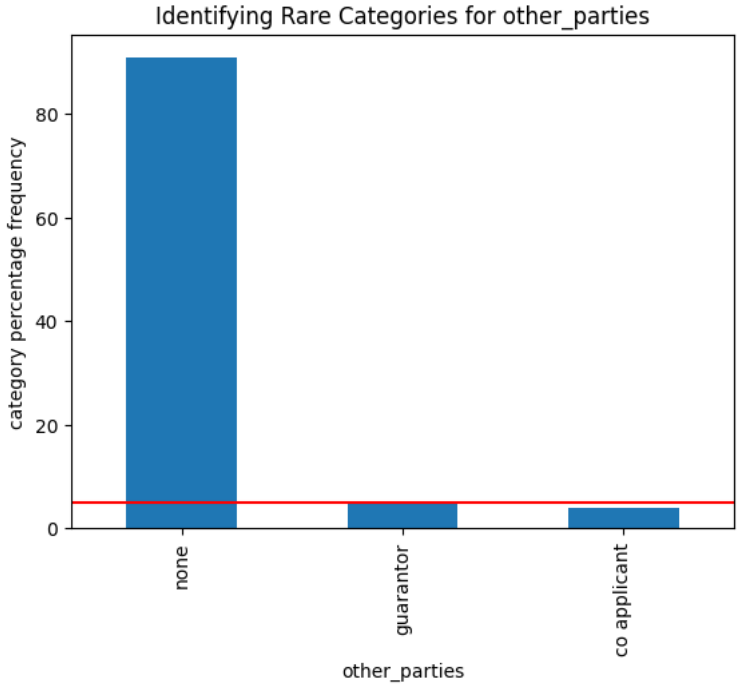

cat
5
0
radio/tv
new car
furniture/equipment
used car
business
education
repairs
domestic appliance
other
retraining
purpose

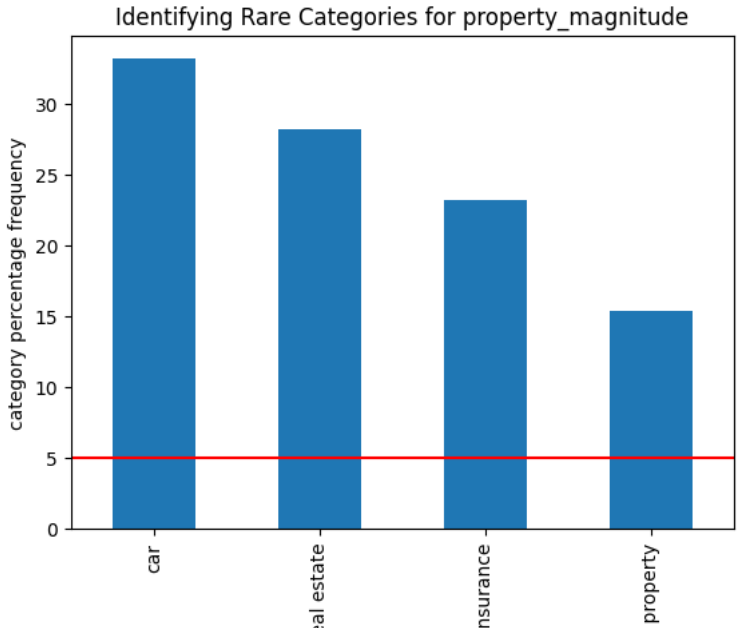

Identifying Rare Categories for savings_status
category percentage frequency
60
50
40
30
20
10
0
<100
no known savings
100<=X<500
500<=X<1000
>=1000
savings_status

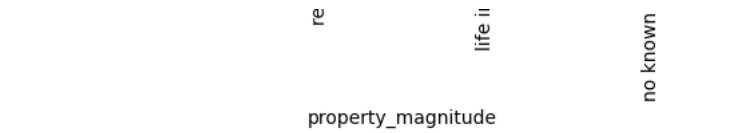

Identifying Rare Categories for employment
category percentage frequency
35
30
25
20
15
10
5
0
1<=X<4
>=7
4<=X<7
<1
unemployed
employment

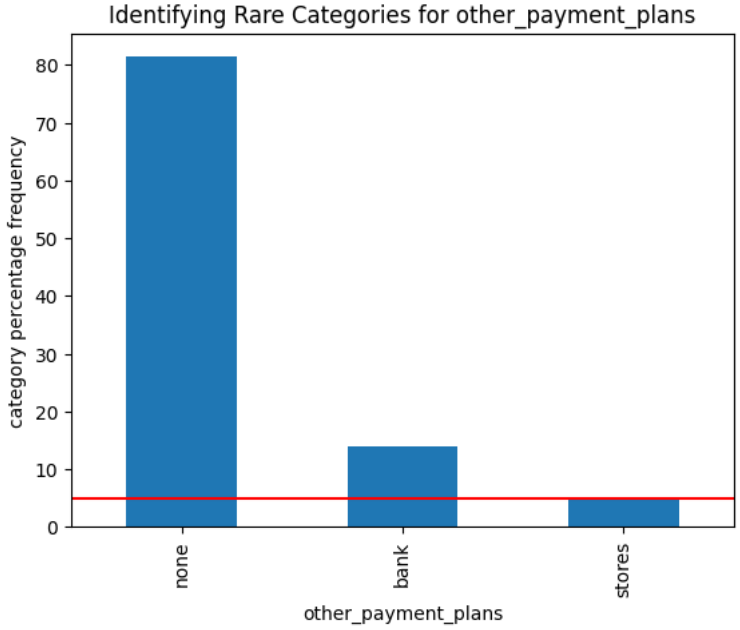

Identifying Rare Categories for personal_status
50

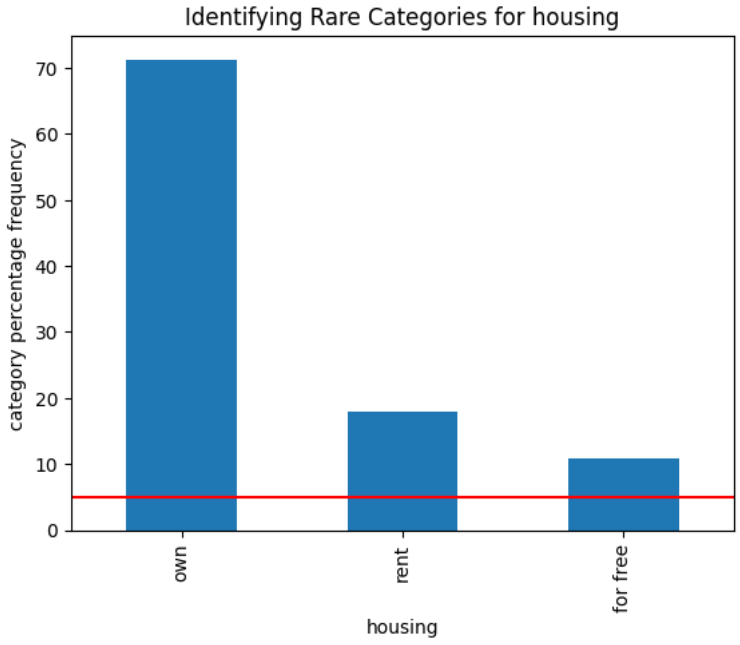

category percentage frequency
0
10
20
30
40
male single
female div/dep/mar
male mar/wid
male div/sep
personal_status

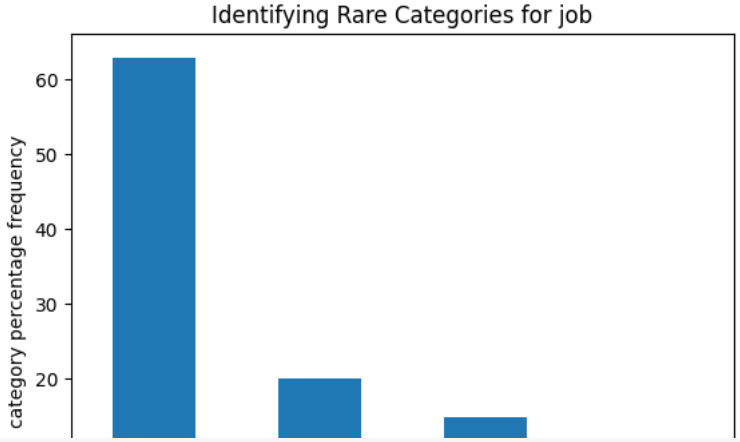

Identifying Rare Categories for other_parties
category percentage frequency
0
20
40
60
80
none
guarantor
co applicant
other_parties


Identifying Rare Categories for property_magnitude
category percentage frequency
0
5
10
15
20
25
30
car
nsurance
property



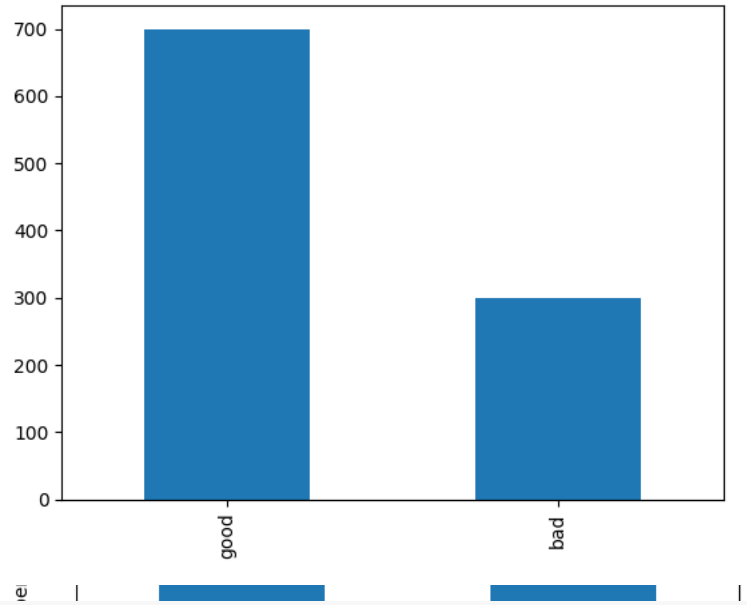




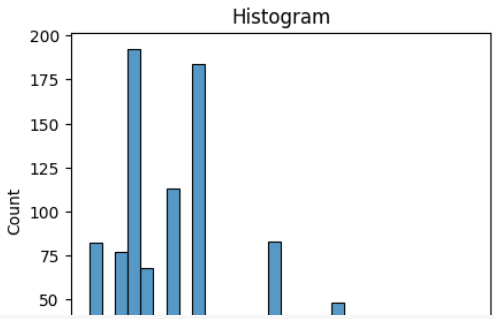


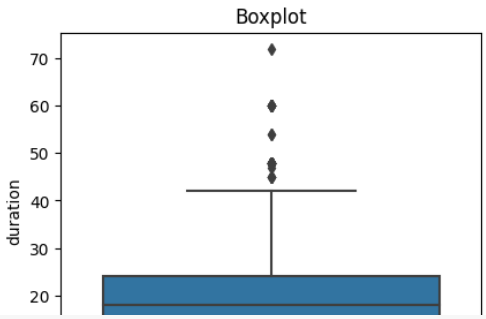

```
b.value_counts().plot.bar()
```

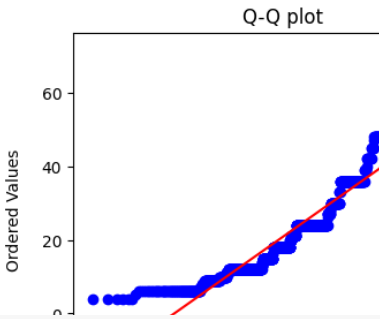
```
<Axes: >
```



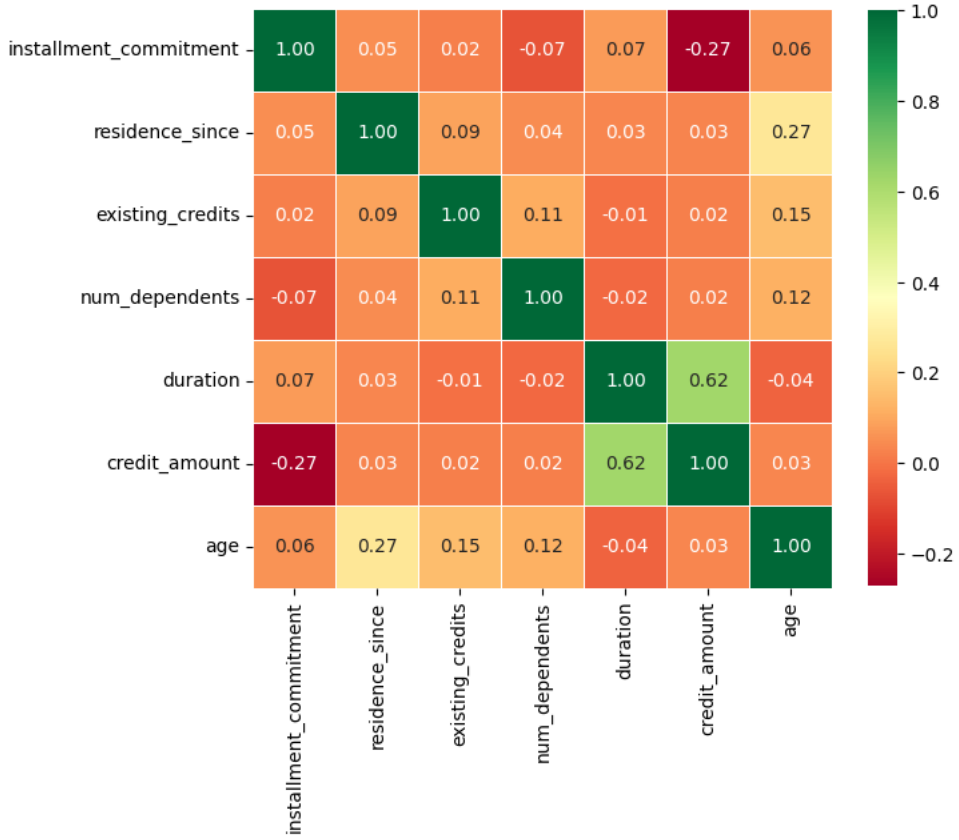

```
for var in continous_1:
    diagnostic_plots(A, var)
```



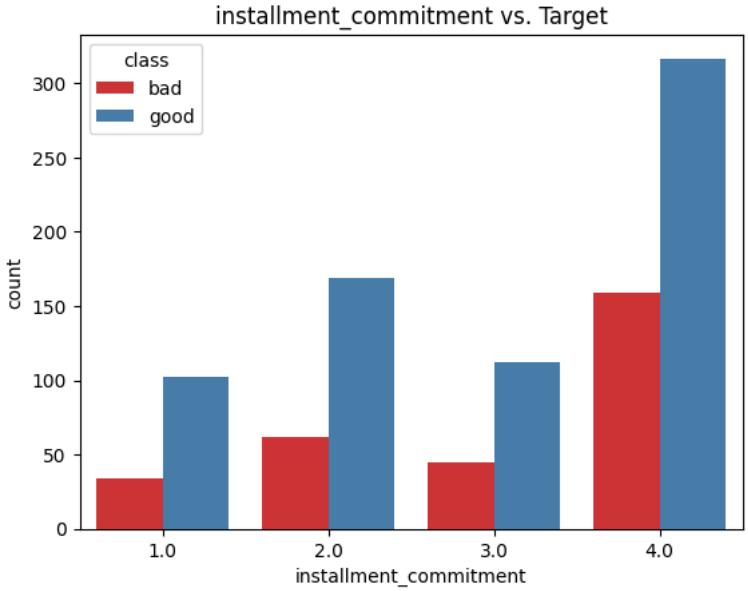


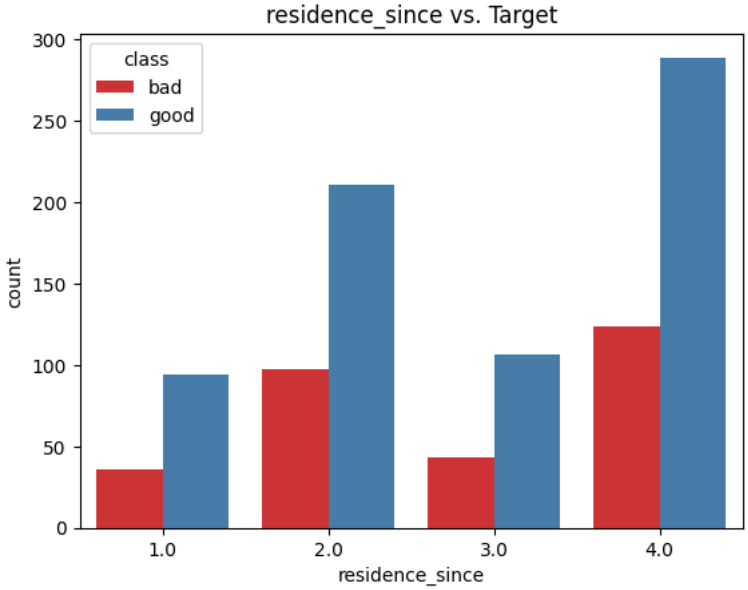


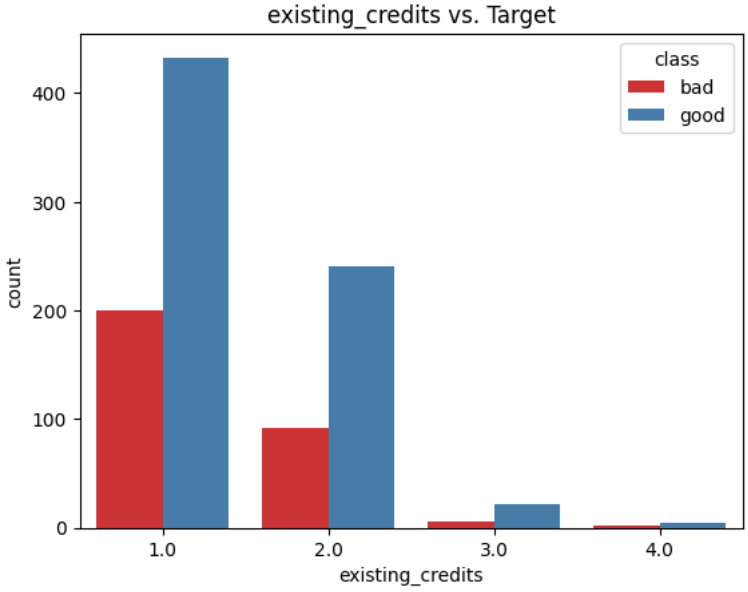

```
corrmat = A[discrete_1 + continous_1].corr().round(2)
top_corr_features = corrmat.index
plt.figure(figsize=(7, 7))
sns.heatmap(A[top_corr_features].corr(),annot=True, square=True, fmt='.2f',
            cbar_kws={"shrink": .80}, linewidths=.5, cmap='RdYlGn');
```

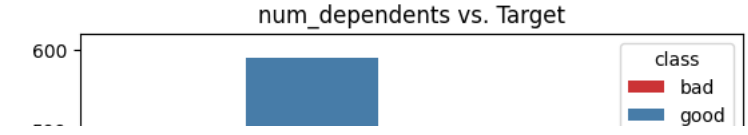



```
for var in categorical_1 and discrete_1:
    sns.countplot(x=var, hue=b, data=A, palette="Set1")
    plt.title(f"{var} vs. Target")
    plt.show()
```


installment_commitment vs. Target
class
bad
good
count
0
50
100
150
200
250
300
1.0
2.0
3.0
4.0
installment_commitment


residence_since vs. Target
class
bad
good
count
0
50
100
150
200
250
300
1.0
2.0
3.0
4.0
residence_since


existing_credits vs. Target
class
bad
good
count
0
100
200
300
400
1.0
2.0
3.0
4.0
existing_credits


num_dependents vs. Target
class
bad
good
600


```
A_train,A_test,b_train,b_test= train_test_split(A,b,test_size=0.33,random_state=0)
```

```
from sklearn.base import BaseEstimator,TransformerMixin

class ConvertToNumpyArray(BaseEstimator,TransformerMixin):
    def __init__(self):
        pass
    def fit(self,X,y=None):
        return self
    def transform(self, X):
        return np.array(X)
```

```
rare_labels_1 = ["foreign_worker","purpose"]
columns_to_transform_1 = ["age","credit_amount","duration"]
```

```
EDA_credit = Pipeline([
    ('rare_label_encoder',RareLabelEncoder(n_categories=1,variables=rare_labels_1,ignore_format=True)),
    ('one_hot_encoder',OneHotEncoder(variables=categorical_1,ignore_format = True)),
    ('yj_transformer',YeoJohnsonTransformer(variables=columns_to_transform_1)),
    ('scaler',MinMaxScaler()),
    ('array_conversion',ConvertToNumpyArray()),
    ('log',LogisticRegression())

])
```

```
param_log = {"log__C":[0.001, 0.01, 0.1, 1, 10, 100, 1000]}
grid_log = GridSearchCV(EDA_credit,param_log,cv=6,return_train_score=True)
```

```
grid_log.fit(A_train,b_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
print(f"best param is : {grid_log.best_params_}")
print(f"CV score is : {grid_log.best_score_}")
```

```
best param is : {'log__C': 0.1}
CV score is : 0.7272727272727274
```

```
print(f"training score ; {grid_log.score(A_train,b_train)}")
print(f"test score: {grid_log.score(A_test,b_test)}")
```

```
training score ; 0.7223880597014926
test score: 0.7818181818181819
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
EDA_credit_2 = Pipeline([
    ('rare_label_encoder',RareLabelEncoder(n_categories=1,variables=rare_labels_1,ignore_format=True)),
    ('one_hot_encoder',OneHotEncoder(variables=categorical_1,ignore_format = True)),
    ('yj_transformer',YeoJohnsonTransformer(variables=columns_to_transform_1)),
    ('scaler',MaxAbsScaler()),
    ('array_conversion',ConvertToNumpyArray()),
    ('log',LogisticRegression())
])
```

```
param_log_2 = {"log__C":[0.001, 0.01, 0.1, 1, 10, 100, 1000]}
grid_log_2 = GridSearchCV(EDA_credit_2,param_log_2,cv=6,return_train_score=True)
```

```
grid_log_2.fit(A_train,b_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
print(f"best param is : {grid_log_2.best_params_}")
print(f"CV score is : {grid_log_2.best_score_}")
```

```
best param is : {'log__C': 1}
CV score is : 0.7597463534963534
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

```
EDA_credit_3 = Pipeline([
    ('rare_label_encoder',RareLabelEncoder(n_categories=1,variables=rare_labels_1,ignore_format=True)),
    ('one_hot_encoder',OneHotEncoder(variables=categorical_1,ignore_format = True)),
    ('log_transformer',LogTransformer(variables=columns_to_transform_1)),
    ('scaler',MaxAbsScaler()),
    ('array_conversion',ConvertToNumpyArray()),
    ('log',LogisticRegression())
])
```

▸ estimator: Pipeline

```
param_log_3 = {"log__C":[1, 10, 100, 1000]}
grid_log_3 = GridSearchCV(EDA_credit_3,param_log_2,cv=6,return_train_score=True)
```

```
grid_log_3.fit(A_train,b_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
print(f"best param is : {grid_log_3.best_params_}")
print(f"CV score is : {grid_log_3.best_score_}")
```

```
best param is : {'log__C': 1}
CV score is : 0.7597463534963534
```

Question2 (7.5 Points) : Linear Regression on Bike Sharing Dataset. OPTIONAL (Try SVM Regression).

- Download the data from following link: https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand'

- Compare KNN (last HW) and Linear Regression. Basd on your anaysis which algorithm you will recommend.
- The aim of the piepline is to predict the rented bike count.

```
"""Importing the required packages"""
!pip install feature_engine -qq
```

```
# For DataFrames and manipulations
import pandas as pd
import numpy as np

# For data Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
%matplotlib inline

# save and load models
import joblib

# Pathlib to navigate file system
from pathlib import Path
import sys

# For splitting the dataset
from sklearn.model_selection import train_test_split
from feature_engine.selection import DropFeatures

# For categorical variables
from feature_engine.encoding import OneHotEncoder
from feature_engine.encoding import RareLabelEncoder

# For scaling the data
from sklearn.preprocessing import StandardScaler

# creating pipelines
from sklearn.pipeline import Pipeline

# Hyper parameter tuning
from sklearn.model_selection import GridSearchCV

# Using KNN classification for our data
from sklearn.neighbors import KNeighborsClassifier

# draws a confusion matrix
from sklearn.metrics import ConfusionMatrixDisplay

# We will use this to download the Dataset
from sklearn.datasets import fetch_openml

# feature engine log transformation
from feature_engine.transformation import LogTransformer

# feature engine wrapper
from feature_engine.wrappers import SklearnTransformerWrapper
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import ShuffleSplit
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
from math import sqrt
```

```
if 'google.colab' in str(get_ipython()):
    from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df = pd.read_csv("/content/drive/MyDrive/Applied_ML/Class_4/Assignment/Datasets/SeoulBikeData.csv", encoding='latin-1')
```

```
df.head(10)
```

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)
0	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0.00	0.0	0.0
1	01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0.00	0.0	0.0
2	01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	0.00	0.0	0.0
3	01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0.00	0.0	0.0
4	01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	0.00	0.0	0.0
5	01/12/2017	100	5	-6.4	37	1.5	2000	-18.7	0.00	0.0	0.0
6	01/12/2017	181	6	-6.6	35	1.3	2000	-19.5	0.00	0.0	0.0

```
categorical = [var for var in df.columns if df[var].dtype == "O" and var not in ["Rented Bike Count"]]
discrete = [var for var in df.columns if df[var].dtype != "O" and len(df[var].unique() < 20) and var not in ["Rented Bike Count"]
continuous = [var for var in df.columns if df[var].dtype != "O" and var not in discrete and var not in ["Rented Bike Count"]]
```

```
X = df.drop(['Rented Bike Count'], axis=1)
y = df["Rented Bike Count"]
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0,test_size=0.33)
```

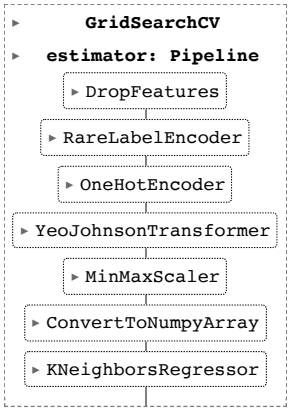
```
from sklearn.base import BaseEstimator,TransformerMixin

class ConvertToNumpyArray(BaseEstimator,TransformerMixin):
    def __init__(self):
        pass
    def fit(self,X,y=None):
        return self
    def transform(self, X):
        return np.array(X)
```

```
columns_to_drop = ['Dew point temperature(°C)']
columns_to_transform = ['Wind speed (m/s)','Rainfall(mm)','Snowfall (cm)','Hour','Solar Radiation (MJ/m2)']
columns_to_scale = ['Hour',
 'Temperature(°C)',
 'Humidity(%)',
 'Wind speed (m/s)',
 'Visibility (10m)',
 'Dew point temperature(°C)',
 'Solar Radiation (MJ/m2)',
 'Rainfall(mm)',
 'Snowfall (cm)']
rare_labels =['Functioning Day']
```

```
from feature_engine.transformation import YeoJohnsonTransformer
from feature_engine.selection.drop_correlated_features import Variables


EDA = Pipeline([
    ('drop_features',DropFeatures(columns_to_drop)),
    ('rare_label_encoder',RareLabelEncoder(n_categories=1,variables=rare_labels,ignore_format=True)),
    ('one_hot_encoder',OneHotEncoder(variables=categorical,ignore_format = True)),
    ('yj_transformer',YeoJohnsonTransformer(variables=columns_to_transform)),
    ('scaler',MinMaxScaler()),
    ('array_conversion',ConvertToNumpyArray()),
    ('knn',KNeighborsRegressor())
])
```

```
param_1 = {'knn__n_neighbors': np.arange(1,11,1)}
grid_knn = GridSearchCV(EDA,param_grid=param_1, cv= 5 , return_train_score=True)
```

```
grid_knn.fit(X_train,y_train)
```




```
grid_knn.best_score_
```

```
0.6840493055684189
```

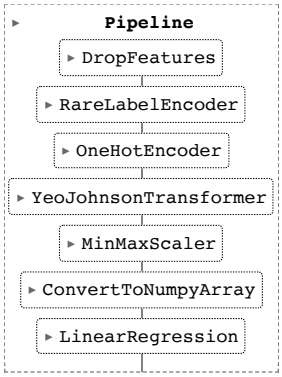
```
grid_knn.score(X_test,y_test)
```

```
0.7253279403212198
```

```
from sklearn.linear_model import Ridge, LinearRegression, Lasso, RidgeCV, LassoCV
```

```
EDA_regression = Pipeline([
    ('drop_features',DropFeatures(columns_to_drop)),
    ('rare_label_encoder',RareLabelEncoder(n_categories=1,variables=rare_labels,ignore_format=True)),
    ('one_hot_encoder',OneHotEncoder(variables=categorical,ignore_format = True)),
    ('yj_transformer',YeoJohnsonTransformer(variables=columns_to_transform)),
    ('scaler',MinMaxScaler()),
    ('array_conversion',ConvertToNumpyArray()),
    ('lr',LinearRegression())
])
```

```
EDA_regression.fit(X_train,y_train)
```




```
from sklearn.model_selection import cross_val_score
```

```
X_train_preds = EDA_regression.predict(X_train)
X_test_preds = EDA_regression.predict(X_test)
print(f'train mse: {mean_squared_error(y_train, X_train_preds)}')
print(f'train rmse: {sqrt(mean_squared_error(y_train, X_train_preds))}')
print(f'train r2: {r2_score(y_train, X_train_preds)}')
print(f'test mse: {mean_squared_error(y_test, X_test_preds)}')
print(f'test rmse: {sqrt(mean_squared_error(y_test, X_test_preds))}')
print(f'test r2: {r2_score(y_test, X_test_preds)}')
scores = cross_val_score(EDA_regression, X_train, y_train,cv=5)
print(scores.mean())
```

```
train mse: 138977.17788379622
train rmse: 372.7964295480795
train r2: 0.6691124375888777
test mse: 154107.2361639571
test rmse: 392.5649451542472
test r2: 0.6219408692476289
0.6181665932497576
```

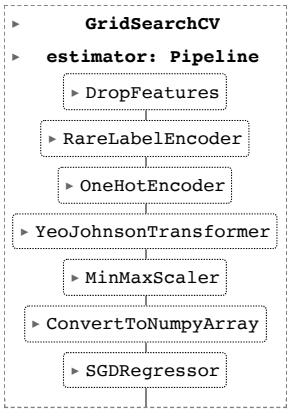
Let me try with SGD regressor

```
from sklearn.linear_model import SGDRegressor
```

```
EDA_SGDregression = Pipeline([
    ('drop_features',DropFeatures(columns_to_drop)),
    ('rare_label_encoder',RareLabelEncoder(n_categories=1,variables=rare_labels,ignore_format=True)),
    ('one_hot_encoder',OneHotEncoder(variables=categorical,ignore_format = True)),
    ('yj_transformer',YeoJohnsonTransformer(variables=columns_to_transform)),
    ('scaler',MinMaxScaler()),
    ('array_conversion',ConvertToNumpyArray()),
    ('SGD', SGDRegressor(max_iter = 1000, tol = 1e-6))
])
```

```
param_sgd = {"SGD__eta0":[0.01, 0.05, 0.1 ,0.5]}

grid_sgd = GridSearchCV(EDA_SGDregression,param_sgd,cv= 6, return_train_score =True, scoring="neg_mean_absolute_error")
```

```
grid_sgd.fit(X_train,y_train)
```




```
X_train_preds_sgd = grid_sgd.predict(X_train)
X_test_preds_sgd = grid_sgd.predict(X_test)
print(f"train mse:  {mean_squared_error(y_train,X_train_preds_sgd)}")
print(f"train rmse : {sqrt(mean_squared_error(y_train,X_train_preds_sgd))}")
print(f"train r2: {r2_score(y_train,X_train_preds_sgd)}")
print(f"test mse : {mean_squared_error(y_test,X_test_preds_sgd)}")
print(f"test rsme : {sqrt(mean_squared_error(y_test,X_test_preds_sgd))}")
print(f"test r2: {r2_score(y_test,X_test_preds_sgd)}")
```

```
train mse:  143242.5688134757
train rmse : 378.4740001816184
train r2: 0.6589570665492366
test mse : 155250.90518865103
test rsme : 394.01891476000367
test r2: 0.6191351962104188
```

Lets Try with Polynomial Regression

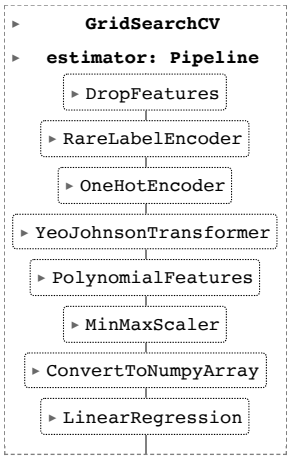
```
from sklearn.preprocessing import PolynomialFeatures
```

```
EDA_Poly = Pipeline([
    ('drop_features',DropFeatures(columns_to_drop)),
    ('rare_label_encoder',RareLabelEncoder(n_categories=1,variables=rare_labels,ignore_format=True)),
    ('one_hot_encoder',OneHotEncoder(variables=categorical,ignore_format = True)),
```

```
    ('yj_transformer',YeoJohnsonTransformer(variables=columns_to_transform)),
    ('Poly', PolynomialFeatures()),
    ('scaler',MinMaxScaler()),
    ('array_conversion',ConvertToNumpyArray()),
    ('lr', LinearRegression())
])
```

```
param_poly = {"Poly__degree":range(1,3)}
grid_poly = GridSearchCV(EDA_Poly,param_poly,cv= 6,n_jobs=1, return_train_score=True)
```

```
grid_poly.fit(X_train,y_train)
```




```
X_train_preds_poly = grid_poly.predict(X_train)
X_test_preds_poly = grid_poly.predict(X_test)
print(f"train mse:  {mean_squared_error(y_train,X_train_preds_poly)}")
print(f"train rmse : {sqrt(mean_squared_error(y_train,X_train_preds_poly))}")
print(f"train r2: {r2_score(y_train,X_train_preds_poly)}")
print(f"test mse : {mean_squared_error(y_test,X_test_preds_poly)}")
print(f"test rsme : {sqrt(mean_squared_error(y_test,X_test_preds_poly))}")
print(f"test r2: {r2_score(y_test,X_test_preds_poly)}")
```

```
train mse:  139210.3794513546
train rmse : 373.1090717891413
train r2: 0.6685572133469209
test mse : 154520.6243514355
test rsme : 393.09111456688447
test r2: 0.6209267366039476
```

Trying Lasso

```
EDA_lasso = Pipeline([
    ('drop_features',DropFeatures(columns_to_drop)),
    ('rare_label_encoder',RareLabelEncoder(n_categories=1,variables=rare_labels,ignore_format=True)),
    ('one_hot_encoder',OneHotEncoder(variables=categorical,ignore_format = True)),
    ('yj_transformer',YeoJohnsonTransformer(variables=columns_to_transform)),
    ('Poly', PolynomialFeatures()),
    ('scaler',MinMaxScaler()),
    ('array_conversion',ConvertToNumpyArray()),
    ('lasso', Lasso(max_iter=1000,tol=1e-06))
])
```

```
param_lasso = {'lasso__alpha':[0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10]}
grid_lasso = GridSearchCV(EDA_lasso,param_lasso,cv=6,return_train_score = True)
```

```
grid_lasso.fit(X_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did n
  model = cd_fast.enet_coordinate_descent(
```

▸ **GridSearchCV**

▸ **estimator: Pipeline**

▸ DropFeatures

▸ RareLabelEncoder

▸ OneHotEncoder

▸ YeoJohnsonTransformer

▸ PolynomialFeatures

▸ MinMaxScaler

```
print(f"Lasso training Score : {grid_lasso.best_score_}")
print(f"Lasso best paramenter : {grid_lasso.best_params_}")
```

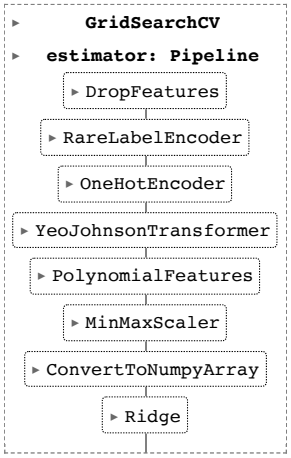
```
Lasso training Score : 0.7724355338426657
Lasso best paramenter : {'lasso__alpha': 0.1}
```

Trying Ridge Regression

```
EDA_Ridge = Pipeline([
    ('drop_features',DropFeatures(columns_to_drop)),
    ('rare_label_encoder',RareLabelEncoder(n_categories=1,variables=rare_labels,ignore_format=True)),
    ('one_hot_encoder',OneHotEncoder(variables=categorical,ignore_format = True)),
    ('yj_transformer',YeoJohnsonTransformer(variables=columns_to_transform)),
    ('Poly', PolynomialFeatures()),
    ('scaler',MinMaxScaler()),
    ('array_conversion',ConvertToNumpyArray()),
    ('ridge', Ridge())
])
```

```
params_ridge = {"ridge__alpha":[0.001, 0.01, 0.1, 1, 10, 100]}
grid_ridge = GridSearchCV(EDA_Ridge,params_ridge, cv= 6, return_train_score=True)
```

```
grid_ridge.fit(X_train,y_train)
```




```
print(f"Best params: {grid_ridge.best_params_}")
print(f"Best score: {grid_ridge.best_score_}")
```

```
Best params: {'ridge__alpha': 1}
Best score: 0.7780464275482287
```

Ridge Regression gives me the Best result than KNN