title: House prices prediction

author: Witawat Muangkot

date: 2022-04-16

Description

this paper will analyze House prices dataset and built model for regression prediction the sale price of houses by following step below:

- Import essential package
- Collect Data
- Split train and test set and take only train and test set
- Cleaning missing data
- Feature name correction
- Category data encoding
  - ordinal encoding
  - one-hot encoding
- data transformation
- Feature Selection
  - correlation feature
  - Lasso
  - RF
  - Boruta
- Feature Extraction
- hyperparameter
- K- cross validation

Import essential package

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'svg'
import os
import sklearn
import sklearn.ensemble
import sklearn.model_selection

from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import VotingRegressor, StackingRegressor
from sklearn.model_selection import KFold, cross_val_score, GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler
from sklearn.preprocessing import PolynomialFeatures
from sklearn.feature_selection import SelectKBest, f_regression, mutual_info_regression
from sklearn.feature_selection import SelectFromModel
from sklearn.feature_selection import RFECV
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

Collect Data

In [2]:
```
df = pd.read_csv("/kaggle/input/house-prices-advanced-regression-techniques/train.csv")
df_test = pd.read_csv("/kaggle/input/house-prices-advanced-regression-techniques/test.csv")
```

In [3]:
```
print("df_train shape:", df.shape) # df_train shape
print("df_test shape:", df_test.shape) # df_test shape
pd.set_option('display.max_columns', 81) # display all columns
pd.set_option('display.max_rows', 81) # display all rows
df.sample(3) # see 3 random samples
```

```
df_train shape: (1460, 81)
df_test shape: (1459, 80)
```

| | Id | MSSubClass | MSZoning | LotFrontage | LotArea | Street | Alley | LotShape | LandContour | Utilities | LotConfig | LandSlope | Neighborhood | Condition1 | Condition2 | BldgType | HouseStyle | OverallQual | OverallCond | YearBuilt | YearRemodAdd | RoofStyle | RoofMatl | Exterior1st | Exterior2nd | MasVnrType | MasVnrArea | ExterQual | ExterCond | Foundation | BsmtQual | BsmtCond | BsmtExposure | BsmtFinType1 | BsmtFinSF1 | BsmtFinType2 | BsmtFinSF2 | BsmtUnfSF | TotalBsmtSF | Heating | HeatingQC | CentralAir | Electrical | 1stFlrSF | 2ndFlrSF | LowQualFinSF | GrLivArea | BsmtFullBath | BsmtHalfBath | FullBath | HalfBath | BedroomAbvGr | KitchenAbvGr | KitchenQual | TotRmsAbvGrd | Functional | Fireplaces | FireplaceQu | GarageType | GarageYrBlt | GarageFinish | GarageCars | GarageArea | GarageQual | GarageCond | PavedDrive | WoodDeckSF | OpenPorchSF | EnclosedPorch | 3SsnPorch | ScreenPorch | PoolArea | PoolQC | Fence | MiscFeature | MiscVal | MoSold | YrSold | SaleType | SaleCondition | SalePrice |
|---|---|---

As the chart above shows that house can sell more for the price after the house with no remodel.

Although the median price for the house with remodel is less than the house with no remodel, but still the house price is higher.

Visualization to see trend of sale prices in period of month, year and with sale condition

In [34]:
```
# Sales variation by month
plt.rcParams['figure.figure-size'] = [8, 6]
plt.scatter(Train['Month'], Train['SalePrice'])
plt.xlabel('Month')
plt.ylabel('Sale Price')
plt.title('House price varies along the year')
plt.show()
```

In [35]:
```
# Sales variation by year
plt.rcParams['figure.figure-size'] = [8, 6]
plt.scatter(Train['YrSold'], Train['SalePrice'])
plt.xlabel('Year')
plt.ylabel('Sale Price')
plt.title('House price varies along the year')
plt.show()
```

In [36]:
```
# Sales variation by Neighborhood
plt.rcParams['figure.figure-size'] = [8, 6]
```

Categorical Encoding

In the dataset has high cardinality and sale details in category features, we will group some of category labels with low frequency into "Other" category.

Then, we will encoding the object columns with number of unique values less than 6 and find the label with low number of frequency less than 5 percent.

In [37]:
```
# Find out the number of unique values
for col in Train.columns:
    if Train[col].dtypes == 'object':
        print('{0:-<40} {1:>4}'.format(col, Train[col].nunique()))
        print(Train[col].value_counts()
        print()

MSZoning---------------------------- 5
RL                  1151
RM                   218
FV                    65
RH                    16
C (all)               10
Name: MSZoning, dtype: int64

Street----------------------------- 2
Pave                1454
Grvl                   6
Name: Street, dtype: int64

Alley------------------------------ 2
Grvl                   50
Pave                   41
Name: Alley, dtype: int64

LotShape--------------------------- 4
Reg                 925
IR1                 484
IR2                  41
IR3                  10
Name: LotShape, dtype: int64

LandContour------------------------ 4
Lvl                 1311
Bnk                   63
HLS                   50
Low                   36
Name: LandContour, dtype: int64

Utilities-------------------------- 2
AllPub              1459
NoSeWa                 1
Name: Utilities, dtype: int64

LotConfig-------------------------- 5
Inside              1052
Corner               263
CulDSac               94
FR2                   47
FR3                    4
Name: LotConfig, dtype: int64

LandSlope-------------------------- 3
Gtl                 1382
Mod                   65
Sev                   13
Name: LandSlope, dtype: int64

Neighborhood----------------------- 25
NAmes               225
CollgCr             150
OldTown             113
Edwards             100
Somerst              86
Gilbert              79
NridgHt              77
Sawyer               74
NWAmes               73
SawyerW              59
BrkSide              58
Crawfor              51
Mitchel              49
NoRidge              41
Timber               38
IDOTRR               37
ClearCr              28
StoneBr              25
SWISU                25
MeadowV              17
Blmngtn              17
BrDale               16
Blueste               2
NPkVill               9
Veenker               11
Name: Neighborhood, dtype: int64

Condition1------------------------- 9
Norm                1260
Feedr                 81
Artery                48
RRAe                  26
PosN                  19
RRAn                  10
PosA                   8
RRNn                   5
Name: Condition1, dtype: int64

Condition2------------------------- 8
Norm                1445
Feedr                 6
Artery                2
PosN                  2
PosA                  1
RRNn                  1
RRAe                  1
RRAn                  1
Name: Condition2, dtype: int64

BldgType--------------------------- 5
1Fam                1220
TwnhsE               114
Duplex               052
Twnhs                043
2fmCon               031
Name: BldgType, dtype: int64

HouseStyle------------------------- 8
1Story              726
2Story              445
1.5Fin              154
SLvl                065
SFoyer              037
1.5Unf              014
2.5Unf              011
2.5Fin              008
Name: HouseStyle, dtype: int64

RoofStyle-------------------------- 6
Gable               1141
Hip                  286
Flat                  013
Gambrel               011
Mansard               007
Shed                  002
Name: RoofStyle, dtype: int64

RoofMatl--------------------------- 8
CompShg             1434
Tar&Grv               011
WdShake               005
WdShngl               005
Metal                 001
Membran               001
Roll                  001
ClayTile              001
Name: RoofMatl, dtype: int64

Exterior1-------------------------- 15
VinylSd             0515
HdBoard             0222
MetalSd             0220
Wd Sdng             0206
Plywood             0108
CemntBd             0061
BrkFace             0050
WdShngl             0056
Stucco              0025
AsbShng             0020
BrkComm             0002
Stone               0002
AsphShn             0002
ImStucc             0001
CBlock              0001
Name: Exterior1st, dtype: int64

Exterior2------------------------- 16
VinylSd             0504
MetalSd             0214
HdBoard             0207
Wd Sdng             0197
Plywood             0142
CmentBd             0060
Wd Shng             0038
Stucco              0026
BrkFace             0025
AsbShng             0020
Brk Cmn             0022
Stone               0005
AsphShn             0004
ImStucc             0001
CBlock              0001
Name: Exterior2nd, dtype: int64

MasVnrType------------------------ 4
None                0864
BrkFace             0445
Stone               0128
BrkCmn              0015
Name: MasVnrType, dtype: int64

ExterQual-------------------------- 4
TA                  0906
Gd                  0488
Ex                  0052
Fa                  0014
Name: ExterQual, dtype: int64

ExterCond-------------------------- 5
TA                  1282
Gd                   146
Fa                    028
Po                    007
Ex                    003
Name: ExterCond, dtype: int64

Foundation------------------------ 6
PConc               0647
CBlock              0634
BrkTil              0146
Slab                0024
Stone               0006
Wood                0003
Name: Foundation, dtype: int64

BsmtQual--------------------------- 4
TA                  0649
Gd                  0618
Ex                  0121
Fa                  0035
Name: BsmtQual, dtype: int64

BsmtCond--------------------------- 4
TA                  1311
Gd                   065
Fa                   045
Po                    002
Name: BsmtCond, dtype: int64

BsmtExposure----------------------- 4
No                  0953
Av                  0221
Gd                  0134
Mn                  0114
Name: BsmtExposure, dtype: int64

BsmtFinType1----------------------- 6
Unf                 0430
GLQ                 0418
ALQ                 0148
BLQ                 0088
Rec                 0054
LwQ                 0037
Name: BsmtFinType1, dtype: int64

BsmtFinType2----------------------- 6
Unf                 1256
Rec                   054
LwQ                   033
BLQ                   023
ALQ                   019
GLQ                   014
Name: BsmtFinType2, dtype: int64

Heating---------------------------- 6
GasA                1428
GasW                   018
Grav                   007
Wall                   004
OthW                   002
Floor                  001
Name: Heating, dtype: int64

HeatingQC-------------------------- 5
Ex                  0741
TA                  0428
Gd                  0241
Fa                  0049
Po                    001
Name: HeatingQC, dtype: int64

CentralAir------------------------ 2
Y                   1365
N                     95
Name: CentralAir, dtype: int64

Electrical------------------------ 5
SBrkr               1334
FuseA                 094
FuseF                 027
FuseP                 003
Mix                   001
Name: Electrical, dtype: int64

KitchenQual------------------------ 4
TA                  0735
Gd                  0586
Ex                  0100
Fa                  0039
Name: KitchenQual, dtype: int64

Functional------------------------ 7
Typ                 1360
Min2                   070
Min1                   065
Mod                    035
Maj1                   019
Maj2                   009
Sev                    001
Name: Functional, dtype: int64

FireplaceQu------------------------ 5
NaN                 0690
Gd                  0380
TA                  0313
Fa                  0033
Po                  0020
Ex                  0028
Name: FireplaceQu, dtype: int64

GarageType------------------------ 6
Attchd              0870
Detchd              0387
BuiltIn             0088
Basment             0019
Calle               0002
2Types              0006
Name: GarageType, dtype: int64

GarageFinish----------------------- 3
Unf                 0605
RFn                 0422
Fin                 0352
Name: GarageFinish, dtype: int64

GarageQual------------------------ 5
TA                  1311
Fa                   048
Gd                   014
Po                   003
Ex                   003
Name: GarageQual, dtype: int64

GarageCond------------------------ 5
TA                  1326
Fa                   035
Gd                   009
Po                   007
Ex                   002
Name: GarageCond, dtype: int64

PavedDrive------------------------ 3
Y                   1340
N                     90
P                     030
Name: PavedDrive, dtype: int64

PoolQC----------------------------- 3
NaN                 1453
Gd                    003
Ex                    002
Fa                    002
Name: PoolQC, dtype: int64

Fence------------------------------ 5
NaN                 1179
MnPrv               0157
GdPrv               0059
GdWo                0054
MnWw                0011
Name: Fence, dtype: int64

MiscFeature------------------------ 4
NaN                 1406
Shed                   049
Othr                   002
Gar2                   002
Name: MiscFeature, dtype: int64

SaleType--------------------------- 9
WD                  1267
New                   122
COD                   043
ConLD                 009
ConLI                 005
ConLw                 005
CWD                   004
Oth                   003
Con                   002
Name: SaleType, dtype: int64

SaleCondition---------------------- 6
Normal              1198
Partial               125
Abnorml               101
Family                020
Alloca                012
AdjLand               004
Name: SaleCondition, dtype: int64
```

In [38]:
```
Cat_cols=['MSZoning','Street','LotShape','LandContour','Utilities','LotConfig','LandSlope','Condition1','Condition2','BldgType','HouseS
```

In [39]:
```
# Group the labels with low frequency into "Other" group
def Cat_cols_encoding(df, cat_cols):
    print('Before: {0}'.format(len(cat_cols)))
    cat_cols = [c for c in cat_cols if(df[c].nunique() > 1)]
    print('After: {0}'.format(len(cat_cols)))
    for c in cat_cols:
        df[c] = df[c].fillna('Other')
    return df, cat_cols

def Cat_encoding(df,threshold, N_cols, OH_cols):
    '''
    : df: pandas dataframe
    : threshold: threshold for grouping
    : N_cols: numerical columns
    : OH_cols: one-hot-encoding columns
    '''
    for c in N_cols:
        tmp = df[c].value_counts(normalize=True).sort_values()
        cum = tmp.cumsum()
        to_replace = list(cum[cum <= threshold].index)
        print('{0:-<40} {1:>4}'.format(c, len(to_replace)))
        df[c] = df[c].replace(to_replace, 'Other')
    return df, N_cols, OH_cols
```

In [40]:
```
Train, N_cols, OH_cols = Cat_encoding(Train, 0.05, Cat_cols)
```

In [41]:
```
Train, Cat_cols = Cat_cols_encoding(Train, Cat_cols)
```

In [42]:
```
# Check the number of unique values
for c in Cat_cols:
    print('{0:-<40} {1:>4}'.format(c, Train[c].nunique()))

MSZoning---------------------------- 5
Street----------------------------- 2
LotShape--------------------------- 4
LandContour------------------------ 4
Utilities-------------------------- 2
LotConfig-------------------------- 5
LandSlope-------------------------- 3
Neighborhood----------------------- 25
Condition1------------------------- 9
Condition2------------------------- 8
BldgType--------------------------- 5
HouseStyle------------------------- 8
RoofStyle-------------------------- 6
RoofMatl--------------------------- 8
Exterior1-------------------------- 15
Exterior2------------------------- 16
MasVnrType------------------------ 4
ExterQual-------------------------- 4
ExterCond-------------------------- 5
Foundation------------------------ 6
BsmtQual--------------------------- 4
BsmtCond--------------------------- 4
BsmtExposure----------------------- 4
BsmtFinType1----------------------- 6
BsmtFinType2----------------------- 6
Heating---------------------------- 6
HeatingQC-------------------------- 5
CentralAir------------------------ 2
Electrical------------------------ 5
KitchenQual------------------------ 4
Functional------------------------ 7
FireplaceQu------------------------ 5
GarageType------------------------ 6
GarageFinish----------------------- 3
GarageQual------------------------ 5
GarageCond------------------------ 5
PavedDrive------------------------ 3
PoolQC----------------------------- 3
Fence------------------------------ 5
MiscFeature------------------------ 4
SaleType--------------------------- 9
SaleCondition---------------------- 6
```

Percentage
Lowfever

```
Mean : 0.700000
Std. Dev. : 0.458831
Min. : 0.000000
Max. : 1.000000
Mode : 1 Boolean, dropped: FALSE

Income
Mean : 0.103044
Std. Dev. : 0.304201
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

Kids
Mean : 0.100041
Std. Dev. : 0.300076
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

LeadSource
Mean : 0.400017
Std. Dev. : 0.489977
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

LastPeriod
Mean : 0.100138
Std. Dev. : 0.300189
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MaritalStatus
Mean : 0.200000
Std. Dev. : 0.400000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalExam
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalHistory
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlan
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanNetwork
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYear
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPrice
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceIDPriceID
Mean : 0.100000
Std. Dev. : 0.300000
Min. : 0.000000
Max. : 1.000000
Mode : 0 Boolean, dropped: FALSE

MedicalPlanYearPriceID
```

Histogram
Probability Plot
Boxplot
Sample
Ordered Values
Theoretical Quantiles
Sample Quantiles

```
[1]: combo_1_boxcox = 1/combo_1[target_name]**0.5
     combo_1_boxcox.name = 'inverse_sqrt_'+target_name
     combo_1_boxcox.describe()
     combo_1_boxcox.to_csv('./data/combo_1_boxcox.csv', index=False)
     combo_1_boxcox.to_pickle('./data/combo_1_boxcox.pkl')
```

```
count    1000.000000
mean        0.000000
std         0.000000
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         0.000000
Name: inverse_sqrt_Cracking_Load, dtype: float64
```

```
[2]: # train_combo_1_boxcox = train_df[train_df.index.isin(train_idx.values)]
     # test_combo_1_boxcox = test_df[test_df.index.isin(test_idx.values)]
     # train_combo_1_boxcox.to_csv('./data/train_combo_1_boxcox.csv')
     # test_combo_1_boxcox.to_csv('./data/test_combo_1_boxcox.csv')
```

```
[3]: for col in X_train.columns:
         print(col, '\n')
         X_combo_1[col] = X_combo_1[col].fillna(0)
         X_combo_1[col] = X_combo_1[col].astype('int')
         X_combo_1[col].value_counts().sort_index()
         X_combo_1[col].value_counts(normalize=True)
```

```
[4]: X_combo_1.shape
     X_combo_1.to_pickle('./data/X_combo_1.pkl')
```

Histogram
Probability Plot
Boxplot
Frequency
Ordered Values
Quantiles
Sample
Sample Quantiles
Theoretical Quantiles
Lifetime Mean
Lifetime Median
Lifetime Mode
Lifetime Variance
Lifetime Skewness
Lifetime Kurtosis
Lifetime Range
Lifetime Interquartile Range
Lifetime Mean Absolute Deviation
Lifetime Coefficient of Variation
Lifetime Median Absolute Deviation
Lifetime Quartile Deviation
Lifetime Coefficient of Quartile Variation
Lifetime Mean Absolute Difference
Lifetime Gini Coefficient
Lifetime Entropy
Lifetime Hartley Mean
Lifetime Quantile Function
Lifetime Moment Generating Function
Lifetime Characteristic Function
Lifetime Cumulant Generating Function
Lifetime Mean Residual Life
Lifetime Vitality Function
Lifetime Hazard Rate Function
Lifetime Cumulative Hazard Rate Function
Lifetime Odds Function
Lifetime Probability Density Function
Lifetime Cumulative Distribution Function
Lifetime Survival Function
Lifetime Probability Mass Function
Lifetime Probability Generating

Feature Selection

Since we have a lot of features that could be predictive, we need to select some features that seem useful for our prediction and remove the ones that are of little or no importance. We will use a combination of a forward methodology by using step_forward_aic with the new score where the impact of model performance

In[ ]:
```
df_train_selected = df_train.clone()
df_test_selected = df_test.clone()

def score(df, target, cv=5):
    '''
    Fast cross validation.
    df - dataset
    target - name of target variable
    cv - number of folds
    '''
    scores = cv * [0.0]
    for i in range(cv):
        train_fold = df.sample(fraction=(1.0 - 1.0/cv), seed=i)
        test_fold = df.join(train_fold, on=df.columns[0], how="anti")
        ...
    return scores
```

In[ ]:
```
score_dataset(df_train, 'y_term', 'AUC')
```

Out[ ]:
```
0.9
```

we'll check the features that contain constant value and will remove it to improve model performance

In[ ]:
```
const_cols = []
for col in df_train.columns:
    if len(df_train[df_train[col] == df_train[col].first()]) == df_train.height:
        const_cols.append(col)
        print(f"{col} - {df_train[col].dtype}")
    ...
const_cols.extend(['poutcome'])
```

Out[ ]:
```
poutcome - str
```

As we notice that there are 15 features that contain the constant value, then we remove those from our dataset

In[ ]:
```
df_train = df_train.drop(const_cols)
df_test = df_test.drop(const_cols)
```

Out[ ]:
```
shape: (36,168, 15)
┌─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┐
│ job_nan ┆ job_ent ┆ job_ret ┆ job_sta ┆ job_sel ┆ job_hou ┆ job_sel ┆ job_ent ┆ job_ret ┆ job_sta ┆ job_sel ┆ job_hou ┆ job_sel ┆ job_ent ┆ job_ret │
│ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     │
│ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      │
└─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┘
shape: (9,045, 15)
┌─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┐
│ job_nan ┆ job_ent ┆ job_ret ┆ job_sta ┆ job_sel ┆ job_hou ┆ job_sel ┆ job_ent ┆ job_ret ┆ job_sta ┆ job_sel ┆ job_hou ┆ job_sel ┆ job_ent ┆ job_ret │
│ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     │
│ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      │
└─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┘
```

The performance after remove some constant feature, a bit increase

check some feature duplicate

In[ ]:
```
# Check for duplicate features
unique_features = set()
duplicate_features = []

for col in df.columns:
    if col == 'y':
        continue
    ...
    if tuple(df[col].sort()) in unique_features:
        print(f"Duplicate feature: {col}")
        duplicate_features.append(col)
    else:
        unique_features.add(tuple(df[col].sort()))

print(len(duplicate_features))
```

Out[ ]:
```
0
```

In[ ]:
```
drop_dupl_cols = []
```

Out[ ]:
```
shape: (9,045, 15)
┌─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┐
│ job_nan ┆ job_ent ┆ job_ret ┆ job_sta ┆ job_sel ┆ job_hou ┆ job_sel ┆ job_ent ┆ job_ret ┆ job_sta ┆ job_sel ┆ job_hou ┆ job_sel ┆ job_ent ┆ job_ret │
│ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     │
│ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      │
└─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┘
```

Check correlation of dataset

In[ ]:
```
corr_matrix = df.corr(method="spearman")
corr_matrix = corr_matrix.fill_nan(0.0).round(2)
corr_matrix = corr_matrix[corr_matrix.columns].round(2)
```

Out[ ]:
```
shape: (15, 15)
```

In[ ]:
```
corr_dataset(df_train, 'y_term', 'PERC')
```

Out[ ]:
```
shape: (15, 3)
┌──────────────┬──────────────┬──────────┐
│ Feature_1    ┆ Feature_2    ┆ Corr     │
│ ---          ┆ ---          ┆ ---      │
│ str          ┆ str          ┆ f64      │
└──────────────┴──────────────┴──────────┘
```

The performance after remove some constant feature, a bit increase

check some feature duplicate

In[ ]:
```
# Check for duplicate features
unique_features = set()
duplicate_features = []

for col in df.columns:
    if col == 'y':
        continue
    ...
    if tuple(df[col].sort()) in unique_features:
        print(f"Duplicate feature: {col}")
        duplicate_features.append(col)
    else:
        unique_features.add(tuple(df[col].sort()))

print(len(duplicate_features))
```

Out[ ]:
```
0
```

In[ ]:
```
drop_dupl_cols = []
```

Out[ ]:
```
shape: (9,045, 15)
┌─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────┐
│ job_nan ┆ job_ent ┆ job_ret ┆ job_sta ┆ job_sel ┆ job_hou ┆ job_sel ┆ job_ent ┆ job_ret ┆ job_sta ┆ job_sel ┆ job_hou ┆ job_sel ┆ job_ent ┆ job_ret │
│ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     ┆ ---     │
│ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      ┆ u8      │
└─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┴─────────┘
```

Check correlation of dataset

In[ ]:
```
corr_matrix = df.corr(method="spearman")
corr_matrix = corr_matrix.fill_nan(0.0).round(2)
corr_matrix = corr_matrix[corr_matrix.columns].round(2)
```

Out[ ]:
```
shape: (15, 15)
```

Now we'll compare the 1-hotted for value frequency and the target rate

- Lower that we'll filter out the feature that it coefficient
- 0% all (Insufficiencies)(Yes and No) filter out the feature that it coefficient
- 100% all (Sufficiencies)(Yes and No) filter out the feature that it coefficient

In[ ]:
```
cat_cols = df.select(pl.col(pl.Enum)).
df_cat = df[cat_cols].
for col in df_cat.columns:
    print(f"{col} - {df_cat[col].value_counts().sort(descending=True)}")
```

Out[ ]:
```
job_nan - shape: (2, 2)
┌─────────┬─────────┐
│ job_nan ┆ count   │
│ ---     ┆ ---     │
│ u8      ┆ u32     │
└─────────┴─────────┘
```

In[ ]:
```
df_dummies = df[cat_cols].to_dummies()
for col in df_dummies.columns:
    print(f"{col} - {df_dummies[col].value_counts().sort(descending=True)}")
```

Out[ ]:
```
job_nan_0 - shape: (2, 2)
┌─────────┬─────────┐
│ job_nan_0 ┆ count   │
│ ---     ┆ ---     │
│ u8      ┆ u32     │
└─────────┴─────────┘
```

In[ ]:
```
# Calculate correlation between dummies and target
target = 'y_term'
for col in df_dummies.columns:
    print(f"{col}: {df_dummies[col].corr(df[target])}")
```

Out[ ]:
```
job_nan_0: nan
job_ent_0: nan
job_ret_0: nan
job_sta_0: nan
job_sel_0: nan
job_hou_0: nan
job_sel
```

{"id":"1","jsonrpc":"2.0","result":{"chain":"mainnet","hash":"0x1b4a9f3c2d7e8b6a5c4d3e2f1a0b9c8d7e6f5a4b3c2d1e0f9a8b7c6d5e4f3a2b","number":"0x1a2b3c4d","parentHash":"0x9e8d7c6b5a4f3e2d1c0b9a8f7e6d5c4b3a2f1e0d9c8b7a6f5e4d3c2b1a0f9e8d","timestamp":"0x65a1b2c3","gasLimit":"0x1c9c380","gasUsed":"0x1312d00","baseFeePerGas":"0x3b9aca00","miner":"0x4838b106fce9647bdf1e7877bf73ce8b0bad5f97","extraData":"0x6e6f6e6365","transactionsRoot":"0x7a3b9c2d4e5f6a1b8c0d9e2f3a4b5c6d7e8f9a0b1c2d3e4f5a6b7c8d9e0f1a2b","receiptsRoot":"0x5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e","stateRoot":"0x2f3e4d5c6b7a8e9d0c1b2a3f4e5d6c7b8a9f0e1d2c3b4a5f6e7d8c9b0a1f2e3d","logsBloom":"0x00000000000

```
[NOTE] Downloaded https://raw.githubusercontent.com/Microsoft/vscode-dotnet-runtime/master/extension/test/unit/Utilities/Resources/Microsoft.Net.Sdk.Publishing.targets to /home/runner/work/_temp/vscode-dotnet-runtime-extension/test/unit/Utilities/Resources/Microsoft.Net.Sdk.Publishing.targets
```

```
[CV] max_features=sqrt, max_depth=None, min_samples_split=5, min_samples_leaf=1, n_estimators=1000 -  1.4min
```