**Functions**

1. Reading and creating train and test data

import pandas as pd

data = pd.read\_csv('AmesHousing.txt', delimiter="\t")

train = data[0:1460]

test = data[1460:]

print(train.info())

target = 'SalePrice'

1. Displaying the data

import matplotlib.pyplot as plt

# For prettier plots.

import seaborn

fig = plt.figure(figsize=(7,15))

ax1 = fig.add\_subplot(3, 1, 1)

ax2 = fig.add\_subplot(3, 1, 2)

ax3 = fig.add\_subplot(3, 1, 3)

train.plot(x="Garage Area", y="SalePrice", ax=ax1, kind="scatter")

train.plot(x="Gr Liv Area", y="SalePrice", ax=ax2, kind="scatter")

train.plot(x="Overall Cond", y="SalePrice", ax=ax3, kind="scatter")

plt.show()

## Finding correlations in variables

train[['Garage Area', 'Gr Liv Area', 'Overall Cond', 'SalePrice']].corr()

1. Using scikit learn

from sklearn.linear\_model import LinearRegression

lr = LinearRegression()

lr.fit(train[['Gr Liv Area']], train['SalePrice'])

print(lr.coef\_)

print(lr.intercept\_)

a0 = lr.intercept\_

a1 = lr.coef\_

1. Checking the accuracy with train and test model

import numpy as np

lr = LinearRegression()

lr.fit(train[['Gr Liv Area']], train['SalePrice'])

from sklearn.metrics import mean\_squared\_error

train\_predictions = lr.predict(train[['Gr Liv Area']])

test\_predictions = lr.predict(test[['Gr Liv Area']])

train\_mse = mean\_squared\_error(train\_predictions, train['SalePrice'])

test\_mse = mean\_squared\_error(test\_predictions, test['SalePrice'])

train\_rmse = np.sqrt(train\_mse)

test\_rmse = np.sqrt(test\_mse)

print(train\_rmse)

print(test\_rmse)

1. Adding more features

cols = ['Overall Cond', 'Gr Liv Area']

lr.fit(train[cols], train['SalePrice'])

train\_predictions = lr.predict(train[cols])

test\_predictions = lr.predict(test[cols])

train\_rmse\_2 = np.sqrt(mean\_squared\_error(train\_predictions, train['SalePrice']))

test\_rmse\_2 = np.sqrt(mean\_squared\_error(test\_predictions, test['SalePrice']))

print(train\_rmse\_2)

print(test\_rmse\_2)

1. Selecting unique features

import pandas as pd

data = pd.read\_csv('AmesHousing.txt', delimiter="\t")

train = data[0:1460]

test = data[1460:]

print(train.info())

## selecting only numerical columns (int and float)

numerical\_train = train.select\_dtypes(include=['int', 'float'])

print(numerical\_train.info())

numerical\_train = numerical\_train.drop(['PID', 'Year Built', 'Year Remod/Add', 'Garage Yr Blt', 'Mo Sold', 'Yr Sold'], axis=1)

null\_series = numerical\_train.isnull().sum()

full\_cols\_series = null\_series[null\_series == 0]

print(full\_cols\_series)

1. Finding correlation with target column

train\_subset = train[full\_cols\_series.index]

corrmat = train\_subset.corr()

sorted\_corrs = corrmat['SalePrice'].abs().sort\_values()

print(sorted\_corrs)

1. Correlation between columns

import seaborn as sns

import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))

strong\_corrs = sorted\_corrs[sorted\_corrs > 0.3]

corrmat = train\_subset[strong\_corrs.index].corr()

sns.heatmap(corrmat)

1. Finding low variance data

unit\_train = train[features]/(train[features].max())

sorted\_vars = unit\_train.var().sort\_values()

print(sorted\_vars)

1. Feature Engineering

Changing categorical columns into numerical ones

text\_cols = df\_no\_mv.select\_dtypes(include=['object']).columns

for col in text\_cols:

print(col+":", len(train[col].unique()))

for col in text\_cols:

train[col] = train[col].astype('category')

train['Utilities'].cat.codes.value\_counts()

## dummy columns

dummy\_cols = pd.DataFrame()

for col in text\_cols:

col\_dummies = pd.get\_dummies(train[col])

train = pd.concat([train, col\_dummies], axis=1)

del train[col]

1. Missing values operations

import pandas as pd

data = pd.read\_csv('AmesHousing.txt', delimiter="\t")

train = data[0:1460]

test = data[1460:]

train\_null\_counts = train.isnull().sum()

df\_missing\_values = train[train\_null\_counts[(train\_null\_counts>0) & (train\_null\_counts<584)].index]

print(df\_missing\_values.isnull().sum())

print(df\_missing\_values.dtypes)

float\_cols = df\_missing\_values.select\_dtypes(include=['float'])

float\_cols = float\_cols.fillna(df\_missing\_values.mean())

print(float\_cols.isnull().sum())