In [ ]:

```
!pip install ipython-autotime
%load_ext autotime
```

In [ ]:

```
!pip install lightgbm
```

In [ ]:

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import KFold
from xgboost import XGBRegressor
from sklearn.model_selection import RandomizedSearchCV,GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,Ridge,Lasso,ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor,BaggingRegressor,ExtraTreesRegressor,A
daBoostRegressor,GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from scipy.stats import skew,boxcox
```

```
time: 9.28 ms
```

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')

%cd /content/drive/My Drive/Applied AI Course/Assignments/23. Self Case Study 1
```

```
Mounted at /content/drive
/content/drive/My Drive/Applied AI Course/Assignments/23. Self Case Study 1
time: 23 s
```

In [ ]:

```
"""
Reading Train and Test data
"""
Train_Data = pd.read_csv('train.csv')
Test_Data = pd.read_csv('test.csv')
```

```
time: 4.58 s
```

In [ ]:

```
r,c = Train_Data.shape   # r -> rows and c  --> columns # 188318 and 130
```

```
time: 1.07 ms
```

In [ ]:

```
y_ = np.log1p(Train_Data['loss'])
Train_Data.drop(['id','loss'], axis=1, inplace=True)
Test_Data.drop(['id'], axis=1, inplace=True)
```

```
Train_Test = pd.concat((Train_Data, Test_Data)).reset_index(drop=True)

Train_Test.head()
```

Out[ ]:

	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	cat10	cat11	cat12	cat13	cat14	cat15	cat16	cat17	cat18	cat19
0	A	B	A	B	A	A	A	A	B	A	B	A	A	A	A	A	A	A	A
1	A	B	A	A	A	A	A	A	B	B	A	A	A	A	A	A	A	A	A
2	A	B	A	A	B	A	A	A	B	B	B	B	B	A	A	A	A	A	A
3	B	B	A	B	A	A	A	A	B	A	A	A	A	A	A	A	A	A	A
4	A	B	A	B	A	A	A	A	B	B	A	B	A	A	A	A	A	A	A

5 rows × 130 columns

```
time: 1.1 s
```

In [ ]:

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
```

```
time: 933 µs
```

In [ ]:

```
categorical_ = [feature for feature in Train_Test.columns if 'cat' in feature]
continuous_ = [feature for feature in Train_Test.columns if 'cont' in feature]

#cat_feature = [n for n in joined.columns if n.startswith('cat')]
#cont_feature = [n for n in joined.columns if n.startswith('cont')]
```

```
time: 2.03 ms
```

In [ ]:

```
for feature in categorical_: Train_Test[feature] = le.fit_transform(Train_Test[feature])
```

```
time: 17.3 s
```

In [ ]:

```
from scipy.stats import skew,boxcox
```

```
time: 1 ms
```

In [ ]:

```
skewed_box = Train_Test.loc[:,"cont1":"cont14"].apply(lambda x: skew(x))
skewed_box
```

Out[ ]:

```
cont1     0.513205
cont2    -0.311146
cont3    -0.007023
cont4     0.417559
cont5     0.679610
cont6     0.458413
cont7     0.825889
cont8     0.673237
cont9     1.067247
cont10    0.352116
cont11    0.281139
cont12    0.291997
cont13    0.376138
cont14    0.250673
dtype: float64
```

```
time: 70.2 ms
```

In [ ]:

```
Train_Test["cont1"], lam = boxcox(Train_Test["cont1"] + 1)
Train_Test["cont2"], lam = boxcox(Train_Test["cont2"] + 1)
Train_Test["cont4"], lam = boxcox(Train_Test["cont4"] + 1)
Train_Test["cont5"], lam = boxcox(Train_Test["cont5"] + 1)
Train_Test["cont6"], lam = boxcox(Train_Test["cont6"] + 1)
Train_Test["cont7"], lam = boxcox(Train_Test["cont7"] + 1)
Train_Test["cont8"], lam = boxcox(Train_Test["cont8"] + 1)
Train_Test["cont9"], lam = boxcox(Train_Test["cont9"] + 1)
Train_Test["cont10"], lam = boxcox(Train_Test["cont10"] + 1)
Train_Test["cont11"], lam = boxcox(Train_Test["cont11"] + 1)
Train_Test["cont12"], lam = boxcox(Train_Test["cont12"] + 1)
Train_Test["cont13"], lam = boxcox(Train_Test["cont13"] + 1)
```

```
time: 6.19 s
```

In [ ]:

```
Train_Data = Train_Test.iloc[:r, :]
Test_Data = Train_Test.iloc[r:, :]
```

```
time: 76.2 ms
```

In [ ]:

```
# split the data into test and train by maintaining same distribution of output varaible
'y_true' [stratify=y_true]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Train_Data, y_,test_size=0.20)
```

```
time: 302 ms
```

XGBoost

In [ ]:

```
xg = XGBRegressor()
```

```
time: 818 µs
```

In [ ]:

```
prams={
    'learning_rate':[0.01,0.03,0.05,0.1,0.15,0.2],
     'n_estimators':[100,200,500,1000,2000],
}
```

```
time: 1.5 ms
```

In [ ]:

```
xgb_grid = GridSearchCV(xg,prams,cv = 3,scoring="neg_mean_squared_error",n_jobs = -1,ver
bose=True)
```

```
time: 1.3 ms
```

In [ ]:

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

```
Fitting 3 folds for each of 30 candidates, totalling 90 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed: 55.1min
[Parallel(n_jobs=-1)]: Done  90 out of  90 | elapsed: 130.7min finished
```

```
[09:25:29] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now dep
```

```
recated in favor of reg:squarederror.
```

Out[ ]:

```
GridSearchCV(cv=3, error_score=nan,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    colsample_bylevel=1, colsample_bynode=1,
                                    colsample_bytree=1, gamma=0,
                                    importance_type='gain', learning_rate=0.1,
                                    max_delta_step=0, max_depth=3,
                                    min_child_weight=1, missing=None,
                                    n_estimators=100, n_jobs=1, nthread=None,
                                    objective='reg:linear', random_state=0,
                                    reg_alpha=0, reg_lambda=1,
                                    scale_pos_weight=1, seed=None, silent=None,
                                    subsample=1, verbosity=1),
             iid='deprecated', n_jobs=-1,
             param_grid={'learning_rate': [0.01, 0.03, 0.05, 0.1, 0.15, 0.2],
                         'n_estimators': [100, 200, 500, 1000, 2000]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='neg_mean_squared_error', verbose=True)
```

```
time: 2h 25min 11s
```

In [ ]:

```
results = pd.DataFrame.from_dict(xgb_grid.cv_results_)
```

```
time: 3.34 ms
```

In [ ]:

```
results.head(2)
```

Out[ ]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_learning_rate	param_n_estimators	params	sp
0	47.502924	0.035372	0.327666	0.006254	0.01	100	{'learning_rate': 0.01, 'n_estimators': 100}	
1	92.062452	0.456628	0.507636	0.008790	0.01	200	{'learning_rate': 0.01, 'n_estimators': 200}	

```
time: 25.7 ms
```

In [ ]:

```
print(xgb_grid.best_params_)
```

```
{'learning_rate': 0.1, 'n_estimators': 2000}
time: 1.32 ms
```

In [ ]:

```
final_xg = XGBRegressor(base_score=0.5, booster='gbtree',
                                    colsample_bylevel=1, colsample_bynode=1,
                                    colsample_bytree=1, gamma=0,
                                    importance_type='gain', learning_rate=0.1,
                                    max_delta_step=0, max_depth=3,
                                    min_child_weight=1, missing=None,
                                    n_estimators=2000, n_jobs=1, nthread=None,
                                    objective='reg:linear', random_state=0,
                                    reg_alpha=0, reg_lambda=1,
                                    scale_pos_weight=1, seed=None, silent=None,
                                    subsample=1, verbosity=1).fit(X_train, y_train)
```

```
[12:53:08] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now dep
```

```
recated in favor of reg:squarederror.
```

In [ ]:

```
y_pred_xg = np.expm1(final_xg.predict(X_test))
y_test_xg = np.expm1(y_test)
print(mean_absolute_error(y_pred_xg, y_test_xg))
```

```
1150.5046796306717
```

GradientBoostingRegressor

In [ ]:

```
GBR = GradientBoostingRegressor()
prams={
    'learning_rate':[0.01,0.03,0.05,0.1,0.15,0.2],
     'n_estimators':[100,200,500,1000,2000],
     'max_depth':[1,2,4],
     'subsample':[.5,.75,1],
     'random_state':[1]
}
```

```
time: 2.26 ms
```

In [ ]:

```
gbr_grid = GridSearchCV(GBR,prams,cv = 3,scoring="neg_mean_squared_error",n_jobs = -1,ve
rbose=True)
```

```
time: 1.2 ms
```

In [ ]:

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

```
Fitting 3 folds for each of 270 candidates, totalling 810 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 40 concurrent workers.
[Parallel(n_jobs=-1)]: Done 120 tasks      | elapsed: 19.2min
[Parallel(n_jobs=-1)]: Done 370 tasks      | elapsed: 79.3min
[Parallel(n_jobs=-1)]: Done 720 tasks      | elapsed: 165.8min
[Parallel(n_jobs=-1)]: Done 810 out of 810 | elapsed: 210.1min finished
```

Out[ ]:

```
GridSearchCV(cv=3, error_score=nan,
             estimator=GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0,
                                                 criterion='friedman_mse',
                                                 init=None, learning_rate=0.1,
                                                 loss='ls', max_depth=3,
                                                 max_features=None,
                                                 max_leaf_nodes=None,
                                                 min_impurity_decrease=0.0,
                                                 min_impurity_split=None,
                                                 min_samples_leaf=1,
                                                 min_samples_split=2,
                                                 min_weight_fraction_leaf=0.0,
                                                 n_estimators=100,
                                                 n_iter_n...
                                                 subsample=1.0, tol=0.0001,
                                                 validation_fraction=0.1,
                                                 verbose=0, warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'learning_rate': [0.01, 0.03, 0.05, 0.1, 0.15, 0.2],
                         'max_depth': [1, 2, 4],
                         'n_estimators': [100, 200, 500, 1000, 2000],
                         'random_state': [1], 'subsample': [0.5, 0.75, 1]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='neg_mean_squared_error', verbose=True)
```

```
time: 4h 47s
```

In [ ]:

```
print(gbr_grid.best_params_)
```

```
{'learning_rate': 0.05, 'max_depth': 4, 'n_estimators': 2000, 'random_state': 1, 'subsamp
le': 0.75}
time: 1.23 ms
```

In [ ]:

```
GBR_final = GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0,
                                      criterion='friedman_mse',
                                      init=None, learning_rate=0.05,
                                      loss='ls', max_depth=4,
                                      max_features=None,
                                      max_leaf_nodes=None,
                                      min_impurity_decrease=0.0,
                                      min_impurity_split=None,
                                      min_samples_leaf=1,
                                      min_samples_split=2,
                                      min_weight_fraction_leaf=0.0,
                                      n_estimators=2000,
                                      subsample=0.75, tol=0.0001,
                                      validation_fraction=0.1,
                                      verbose=0, warm_start=False)
```

```
time: 3.66 ms
```

In [ ]:

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

Out[ ]:

```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.05, loss='ls', max_depth=4,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=2000,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=0.75, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

```
time: 32min 35s
```

In [ ]:

```
y_pred_xbr = np.expm1(GBR_final.predict(X_test))
y_test_xbr = np.expm1(y_test)
print(mean_absolute_error(y_pred_xbr, y_test_xbr))
```

```
1147.8637829161644
time: 2.27 s
```

lightgbm

In [ ]:

```
from lightgbm import LGBMRegressor
```

In [ ]:

```
params = {
 'boosting_type': ['gbdt','dart','goss','rf'],
 'metric': ['l2', 'l1'],
 'min_child_weight': [1e-5, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4],
 'learning_rate': [0.01,0.03,0.05,0.1,0.15,0.2],
 'reg_alpha': [1e-2, 1e-1, 1, 1e1, 1e2],
 'reg_lambda': [1e-2, 1e-1, 1, 1e1, 1e2],
}
```

```
time: 2 ms
```

In [ ]:

```
lgb_r = LGBMRegressor(max_depth=-1, random_state=314, n_estimators=5000)
```

```
time: 856 µs
```

In [ ]:

```
lgbmr_rs = RandomizedSearchCV(lgb_r,params,cv = 3,scoring="neg_mean_squared_error",n_job
s = -1,verbose=True)
```

```
time: 929 µs
```

In [ ]:

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

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 40 concurrent workers.
[Parallel(n_jobs=-1)]: Done  13 out of  30 | elapsed:  4.2min remaining:  5.4min
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed: 40.6min finished
```

Out[ ]:

```
RandomizedSearchCV(cv=3, error_score=nan,
                   estimator=LGBMRegressor(boosting_type='gbdt',
                                           class_weight=None,
                                           colsample_bytree=1.0,
                                           importance_type='split',
                                           learning_rate=0.1, max_depth=-1,
                                           min_child_samples=20,
                                           min_child_weight=0.001,
                                           min_split_gain=0.0,
                                           n_estimators=5000, n_jobs=-1,
                                           num_leaves=31, objective=None,
                                           random_state=314, reg_alpha=0.0,
                                           reg_lambda=0.0, silen...
                   param_distributions={'boosting_type': ['gbdt', 'dart',
                                                          'goss', 'rf'],
                                        'learning_rate': [0.01, 0.03, 0.05, 0.1,
                                                          0.15, 0.2],
                                        'metric': ['l2', 'l1'],
                                        'min_child_weight': [1e-05, 0.001, 0.01,
                                                             0.1, 1, 10.0,
                                                             100.0, 1000.0,
                                                             10000.0],
                                        'reg_alpha': [0.01, 0.1, 1, 10.0,
                                                      100.0],
                                        'reg_lambda': [0.01, 0.1, 1, 10.0,
                                                       100.0]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring='neg_mean_squared_error',
                   verbose=True)
```

```
time: 42min 44s
```

In [ ]:

```
print(lgbmr_rs.best_params_)
```

```
{'reg_lambda': 100.0, 'reg_alpha': 10.0, 'min_child_weight': 0.01, 'metric': 'l1', 'learn
ing_rate': 0.03, 'boosting_type': 'gbdt'}
time: 1.95 ms
```

In [ ]:

```
estimator=LGBMRegressor(boosting_type='gbdt',class_weight=None,colsample_bytree=1.0,impor
tance_type='split',learning_rate=0.03, max_depth=-1,
                        min_child_samples=20,min_child_weight=0.001,min_split_gain=0.0,n
_estimators=5000, n_jobs=-1,num_leaves=31, objective=None,
                         random_state=314, reg_alpha=10.0, reg_lambda=100.0,metric= 'l1'
).fit(X_train, y_train)
```

```
time: 45.3 s
```

In [ ]:

```
y_pred_lgm = np.expm1(estimator.predict(X_test))
y_test_lgm = np.expm1(y_test)
print(mean_absolute_error(y_pred_lgm, y_test_lgm))
```

```
1138.9519112163346
time: 625 ms
```