Sameeullah_File1_HW6

December 21, 2021

1 HW6: Required Submissions:

1. Submit colab/jupyter notebooks and pdf files.
2. For this HW, you will use XGBoost Regressiona and Random Forest regreesion on Bike Sharing Dataset.
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. While choosing the pre-processing step, keep in mind the algorithm that you are using. .
5. Pdf version of the notebooks (HWs will not be graded if pdf version is not provided).
6. **The notebooks and pdf files should have the output.**
7. **Name files as follows : FirstName_file1_hw6, FirstName_file2_h6**

2 Question (10 Points) : XGBoost Regression and Random Forest regression on Bike Sharing Dataset

- Download the data from following link: https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+De
- Your goal is to predict the rented bike count.
- Craete a separate pipeline for each algorithm.
- Compare KNN (last HW), DecisonTree and Linear Regression, XgBoost, and Random Forest Regression. Basd on your anaysis which algorithm you will recommend.
- For XGBoost Notebook also provide a list and explanation of different hyperparameters availaible.

[1]:
```
import warnings
warnings.filterwarnings(action='once')
```

[2]:
```
%%capture
!pip install feature_engine
```

[3]:
```
%%capture
!pip install -U scikit-learn
```

[4]:
```
import sklearn
import feature_engine
```

```
/usr/lib/python3.7/importlib/_bootstrap.py:219: RuntimeWarning: numpy.ufunc size
changed, may indicate binary incompatibility. Expected 192 from C header, got
216 from PyObject
  return f(*args, **kwds)
/usr/lib/python3.7/importlib/_bootstrap.py:219: RuntimeWarning: numpy.ufunc size
changed, may indicate binary incompatibility. Expected 192 from C header, got
216 from PyObject
  return f(*args, **kwds)
/usr/lib/python3.7/importlib/_bootstrap.py:219: RuntimeWarning: numpy.ufunc size
changed, may indicate binary incompatibility. Expected 192 from C header, got
216 from PyObject
  return f(*args, **kwds)
```

[5]:
```
# For DataFrames and manipulations
import pandas as pd
import numpy as np
import scipy.stats as stats

# For data Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.offline as po
import plotly.graph_objects as go

%matplotlib inline
import plotly.io as pio
pio.renderers.default = 'colab'

# For splitting the dataset
from sklearn.model_selection import train_test_split

# drop arbitrary features
from sklearn.datasets import fetch_openml

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

# For scaling the data
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

from feature_engine.transformation import YeoJohnsonTransformer
from feature_engine.transformation import LogTransformer
```

```
# DIscretization
from sklearn.preprocessing import KBinsDiscretizer

# Handling Outliers
from feature_engine.outliers import Winsorizer

# feature engine wrapper
from feature_engine.wrappers import SklearnTransformerWrapper

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

# creating pipelines
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

# Hyper parameter tuning
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold

# learning Curves
from sklearn.model_selection import learning_curve

# draws a confusion matrix
from sklearn.metrics import plot_confusion_matrix
from scipy.stats import uniform, truncnorm, randint, loguniform


# save and load models
import joblib

# Pathlib to navigate file system
from pathlib import Path

from sklearn.neighbors import KNeighborsRegressor



#import os
#os.makedirs("/content/drive/MyDrive/teaching_fall_2021/ml_fall_2021/
 ↪HW_Assignments/HW6/saved_models")
#!ls
```

```
/usr/lib/python3.7/importlib/_bootstrap.py:219: RuntimeWarning: numpy.ufunc size
changed, may indicate binary incompatibility. Expected 192 from C header, got
216 from PyObject
  return f(*args, **kwds)
```

```
/usr/local/lib/python3.7/dist-packages/jsonschema/compat.py:6:
DeprecationWarning:

Using or importing the ABCs from 'collections' instead of from 'collections.abc'
is deprecated since Python 3.3,and in 3.9 it will stop working

/usr/lib/python3.7/importlib/_bootstrap.py:219: RuntimeWarning:

numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from
C header, got 216 from PyObject
```

[6]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

[7]:
```
save_model_folder  = Path('/content/drive/MyDrive/teaching_fall_2021/
 ↪ml_fall_2021/HW_Assignments/HW6/saved_models')
```

[8]:
```
# Load data from  https://archive.ics.uci.edu/ml/datasets/
 ↪Seoul+Bike+Sharing+Demand'
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00560/
 ↪SeoulBikeData.csv'
!wget {url} -P {'/content/drive/MyDrive/teaching_fall_2021/ml_fall_2021/
 ↪HW_Assignments/HW6/'}

datafolder = Path('/content/drive/MyDrive/teaching_fall_2021/ml_fall_2021/
 ↪HW_Assignments/HW6/')
```

```
--2021-12-21 19:14:26--  https://archive.ics.uci.edu/ml/machine-learning-
databases/00560/SeoulBikeData.csv
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 604166 (590K) [application/x-httpd-php]
Saving to: /content/drive/MyDrive/teaching_fall_2021/ml_fall_2021/HW_Assignment
s/HW6/SeoulBikeData.csv.9

SeoulBikeData.csv.9 100%[===================>] 590.01K  1.50MB/s    in 0.4s

2021-12-21 19:14:26 (1.50 MB/s) - /content/drive/MyDrive/teaching_fall_2021/ml_
fall_2021/HW_Assignments/HW6/SeoulBikeData.csv.9 saved [604166/604166]
```

[9]:
```
bike_data = datafolder /'SeoulBikeData.csv'
#X, y = fetch_openml("credit-g", version=1, as_frame=True, return_X_y=True)

with open(bike_data, encoding="utf8",errors='ignore') as csv_file:
    df = pd.read_csv(csv_file)
df.head()
```

[9]:
```
         Date  Rented Bike Count  Hour  ...  Seasons     Holiday  Functioning
Day
0  01/12/2017                254     0  ...   Winter  No Holiday
Yes
1  01/12/2017                204     1  ...   Winter  No Holiday
Yes
2  01/12/2017                173     2  ...   Winter  No Holiday
Yes
3  01/12/2017                107     3  ...   Winter  No Holiday
Yes
4  01/12/2017                 78     4  ...   Winter  No Holiday
Yes

[5 rows x 14 columns]
```

[10]:
```
y = df.iloc[:,1]
print(y.head())
x=df.iloc[:,2:]
x.head()
```

```
0    254
1    204
2    173
3    107
4     78
Name: Rented Bike Count, dtype: int64
```

[10]:
```
   Hour  Temperature(C)  Humidity(%)  ...  Seasons     Holiday  Functioning Day
0     0            -5.2           37  ...   Winter  No Holiday              Yes
1     1            -5.5           38  ...   Winter  No Holiday              Yes
2     2            -6.0           39  ...   Winter  No Holiday              Yes
3     3            -6.2           40  ...   Winter  No Holiday              Yes
4     4            -6.0           36  ...   Winter  No Holiday              Yes

[5 rows x 12 columns]
```

[11]:
```
def plot_learning_curve(estimator, title, X, y, axes=None, ylim=None, cv=None,
                        n_jobs=None, train_sizes=np.linspace(.2, 1.0, 5)):
    """
    Generate 2 plots: the test and training learning curve, the training
    samples vs fit times curve.
```

```
    Parameters
    ----------
    estimator : estimator instance
        An estimator instance implementing `fit` and `predict` methods which
        will be cloned for each validation.

    title : str
        Title for the chart.

    X : array-like of shape (n_samples, n_features)
        Training vector, where ``n_samples`` is the number of samples and
        ``n_features`` is the number of features.

    y : array-like of shape (n_samples) or (n_samples, n_features)
        Target relative to ``X`` for classification or regression;
        None for unsupervised learning.

    axes : array-like of shape (3,), default=None
        Axes to use for plotting the curves.

    ylim : tuple of shape (2,), default=None
        Defines minimum and maximum y-values plotted, e.g. (ymin, ymax).

    cv : int, cross-validation generator or an iterable, default=None
        Determines the cross-validation splitting strategy.
        Possible inputs for cv are:

          - None, to use the default 5-fold cross-validation,
          - integer, to specify the number of folds.
          - :term:`CV splitter`,
          - An iterable yielding (train, test) splits as arrays of indices.

        For integer/None inputs, if ``y`` is binary or multiclass,
        :class:`StratifiedKFold` used. If the estimator is not a classifier
        or if ``y`` is neither binary nor multiclass, :class:`KFold` is used.

        Refer :ref:`User Guide <cross_validation>` for the various
        cross-validators that can be used here.

    n_jobs : int or None, default=None
        Number of jobs to run in parallel.
        ``None`` means 1 unless in a :obj:`joblib.parallel_backend` context.
        ``-1`` means using all processors. See :term:`Glossary <n_jobs>`
        for more details.

    train_sizes : array-like of shape (n_ticks,)
        Relative or absolute numbers of training examples that will be used to
```

```
        generate the learning curve. If the ``dtype`` is float, it is regarded
        as a fraction of the maximum size of the training set (that is
        determined by the selected validation method), i.e. it has to be within
        (0, 1]. Otherwise it is interpreted as absolute sizes of the training
        sets. Note that for classification the number of samples usually have
        to be big enough to contain at least one sample from each class.
        (default: np.linspace(0.1, 1.0, 5))
    """
    if axes is None:
        _, axes = plt.subplots(1, 2, figsize=(10, 5))

    axes[0].set_title(title)
    if ylim is not None:
        axes[0].set_ylim(*ylim)
    axes[0].set_xlabel("Training examples")
    axes[0].set_ylabel("Score")

    train_sizes, train_scores, test_scores, fit_times, _ = \
        learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs,
                       train_sizes=train_sizes,
                       return_times=True,
                       random_state=123)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    fit_times_mean = np.mean(fit_times, axis=1)
    fit_times_std = np.std(fit_times, axis=1)

    # Plot learning curve
    axes[0].grid()
    axes[0].fill_between(train_sizes, train_scores_mean - train_scores_std,
                         train_scores_mean + train_scores_std, alpha=0.1,
                         color="r")
    axes[0].fill_between(train_sizes, test_scores_mean - test_scores_std,
                         test_scores_mean + test_scores_std, alpha=0.1,
                         color="g")
    axes[0].plot(train_sizes, train_scores_mean, 'o-', color="r",
                 label="Training score")
    axes[0].plot(train_sizes, test_scores_mean, 'o-', color="g",
                 label="Cross-validation score")
    axes[0].legend(loc="best")

    # Plot n_samples vs fit_times
    axes[1].grid()
    axes[1].plot(train_sizes, fit_times_mean, 'o-')
    axes[1].fill_between(train_sizes, fit_times_mean - fit_times_std,
```

```
                         fit_times_mean + fit_times_std, alpha=0.1)
    axes[1].set_xlabel("Training examples")
    axes[1].set_ylabel("fit_times")
    axes[1].set_title("Scalability of the model")

    return plt
```

[12]:
```
# Create a list of categorical variables
# Since the dtype of categorical variable is Object we can compare the values␣
 ↪with 'O'
categorical = [var for var in x.columns if x[var].dtype.name == 'category']

# Create a list of discrete variables
# we do not want to consider Exited as this is target variable
discrete = [
    var for var in x.columns if x[var].dtype.name != 'category'
    and len(x[var].unique()) < 20
]

# Create a list of continuous Variables
continuous = [
    var for var in x.columns if x[var].dtype.name != 'category'
    if var not in discrete
]
```

[13]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4,␣
 ↪random_state=123,stratify =x[['Functioning Day','Holiday']])
x_train.head()
```

[13]:
```
      Hour  Temperature(C)  Humidity(%)  ...  Seasons     Holiday  Functioning
Day
2973    21            17.0           83  ...   Spring  No Holiday
Yes
2926    22            16.7           83  ...   Spring  No Holiday
Yes
340      4            -7.1           59  ...   Winter  No Holiday
Yes
8668     4             3.1           88  ...   Autumn  No Holiday
Yes
2985     9            10.4           81  ...   Spring  No Holiday
Yes

[5 rows x 12 columns]
```

[14]:
```
bikepipeline = Pipeline([
            #('rare labels', RareLabelEncoder(tol=.05,n_categories =2,variables␣
 ↪= ['Holiday','Functioning Day'])),
            ('ohe',OneHotEncoder(variables=categorical,drop_last=True)),
            #('log_transform',LogTransformer(variables=continuous)),
```

```
            ('scalar', SklearnTransformerWrapper(StandardScaler(), variables =␣
 ↪continuous)),
            ('knn_reg',KNeighborsRegressor())
])
```

[15]:
```
param_grid_1 = {
    'scalar__transformer': [StandardScaler()],
    'knn_reg__n_neighbors': range(2,20,2),
    'knn_reg__weights': ['uniform', 'distance'],
    'knn_reg__p': [1, 2],
    'knn_reg__n_jobs':[-1]
}

grid_knn_1 = GridSearchCV(bikepipeline, param_grid_1,cv=5,return_train_score=␣
 ↪True)
```

[16]:
```
grid_knn_1.fit(x_train,y_train)
```

[16]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('ohe',
                                        OneHotEncoder(drop_last=True,
                                                      variables=[])),
                                       ('scalar',
SklearnTransformerWrapper(transformer=StandardScaler(),
variables=['Hour',
'Temperature(C)',
'Humidity(%)',
'Wind '
'speed '
'(m/s)',
'Visibility '
'(10m)',
'Dew '
'point '
'temperature(C)',
'Solar '
'Radiation '
'(MJ/m2)',
'Rainfall(mm)',
'Snowfall '
'(cm)'])),
                                       ('knn_reg', KNeighborsRegressor())]),
             param_grid={'knn_reg__n_jobs': [-1],
                         'knn_reg__n_neighbors': range(2, 20, 2),
                         'knn_reg__p': [1, 2],
                         'knn_reg__weights': ['uniform', 'distance'],
                         'scalar__transformer': [StandardScaler()]},
             return_train_score=True)
```

[17]:
```
grid_knn_1.best_params_
```

[17]:
```
{'knn_reg__n_jobs': -1,
 'knn_reg__n_neighbors': 4,
 'knn_reg__p': 1,
 'knn_reg__weights': 'distance',
 'scalar__transformer': StandardScaler()}
```

[18]:
```
# Here  save_model_folder is folder where I have saved models. Change that to␣
 ↪appropriate location.
# This variable is defined in section Mount Google Drive, Import Data

# specify the file to save the best estimator
file_best_estimator_round1 = save_model_folder / 'knn_reg_round1_best_estimator.
 ↪pkl'

# specify the file to save complete grid results
file_complete_grid_round1 = save_model_folder / 'knn_reg_round1_complete_grid.
 ↪pkl'
```

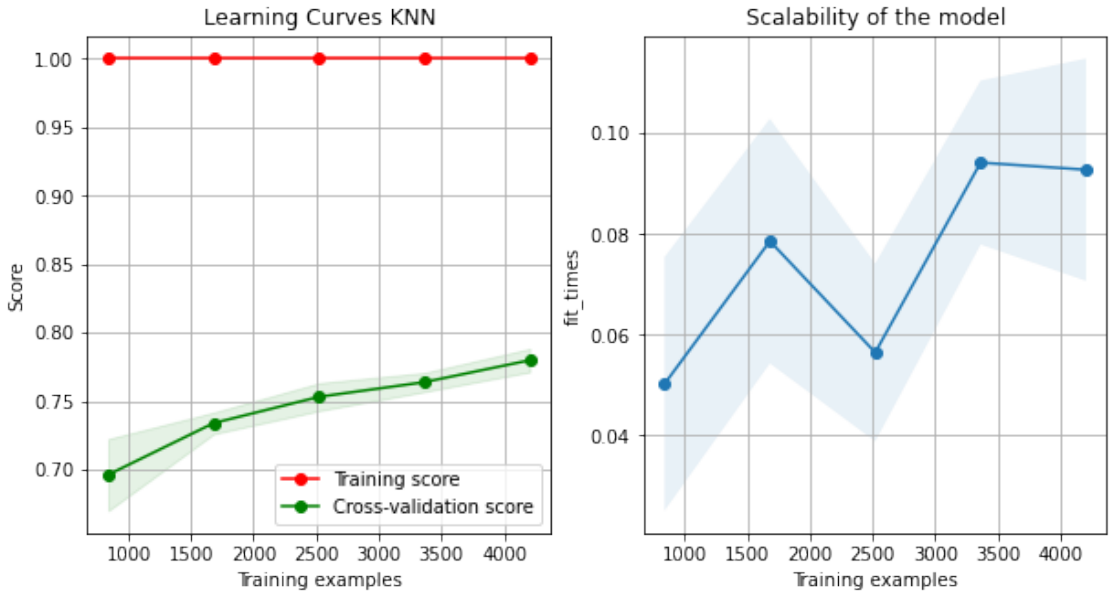
[19]:
```
# save the best estimator
joblib.dump(grid_knn_1.best_estimator_, file_best_estimator_round1)

# save complete grid results
joblib.dump(grid_knn_1, file_complete_grid_round1)
```

[19]:
```
['/content/drive/MyDrive/teaching_fall_2021/ml_fall_2021/HW_Assignments/HW6/save
d_models/knn_reg_round1_complete_grid.pkl']
```

[20]:
```
# load  the best estimator
loaded_best_estimator_round1 = joblib.load(file_best_estimator_round1)

# load complete grid results
loaded_complete_grid_round1 = joblib.load(file_complete_grid_round1)
```

[21]:
```
# plot learning curves
# Notice that we are using the best estimator
plot_learning_curve(loaded_best_estimator_round1, 'Learning Curves KNN',␣
 ↪x_train, y_train, n_jobs=-1)
```

[21]:
```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.7/dist-
packages/matplotlib/pyplot.py'>
```



[22]:
```
print(loaded_best_estimator_round1.score(x_train,y_train))

print(loaded_complete_grid_round1.best_score_)
```

```
1.0
0.7799892227499619
```

[23]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LogisticRegression
from feature_engine.encoding import DecisionTreeEncoder
from sklearn.tree import DecisionTreeClassifier
```

[24]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.4,␣
↪random_state=123, stratify =x[['Functioning Day','Holiday']])
```

[25]:
```
churn_pipeline_dtree = Pipeline([
    ('one_hot_encoder',
      OneHotEncoder(variables=categorical,drop_last=True)),

    ('dtree',
     DecisionTreeClassifier(random_state=0))
])
```

[26]:
```
param_grid__tree_1 = {
    'dtree__max_depth': np.arange(4,20),
    'dtree__min_samples_leaf': np.arange(2,20,4)
    #'dtree__max_leaf_nodes': np.arange(4, 20)
     }
```

[27]:
```
grid_dtree1 = GridSearchCV(churn_pipeline_dtree, param_grid__tree_1,
                           cv=5, return_train_score= True, n_jobs=-1)
grid_dtree1.fit(X_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680:
UserWarning:

The least populated class in y has only 1 members, which is less than
n_splits=5.
```

[27]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('one_hot_encoder',
                                        OneHotEncoder(drop_last=True,
                                                      variables=[])),
                                       ('dtree',
 DecisionTreeClassifier(random_state=0))]),
             n_jobs=-1,
             param_grid={'dtree__max_depth': array([ 4,  5,  6,  7,  8,  9, 10,
 11, 12, 13, 14, 15, 16, 17, 18, 19]),
                         'dtree__min_samples_leaf': array([ 2,  6, 10, 14,
 18])},
             return_train_score=True)
```

[28]:
```
print(grid_dtree1.best_params_)
```

```
{'dtree__max_depth': 15, 'dtree__min_samples_leaf': 2}
```

[29]:
```
file_params_tree1 = save_model_folder / 'dtree_round1_params.pkl'
file_model_tree1 = save_model_folder / 'dtree_round1_model.pkl'
```

[30]:
```
joblib.dump(grid_dtree1.best_estimator_, file_params_tree1)
joblib.dump(grid_dtree1, file_model_tree1)
```

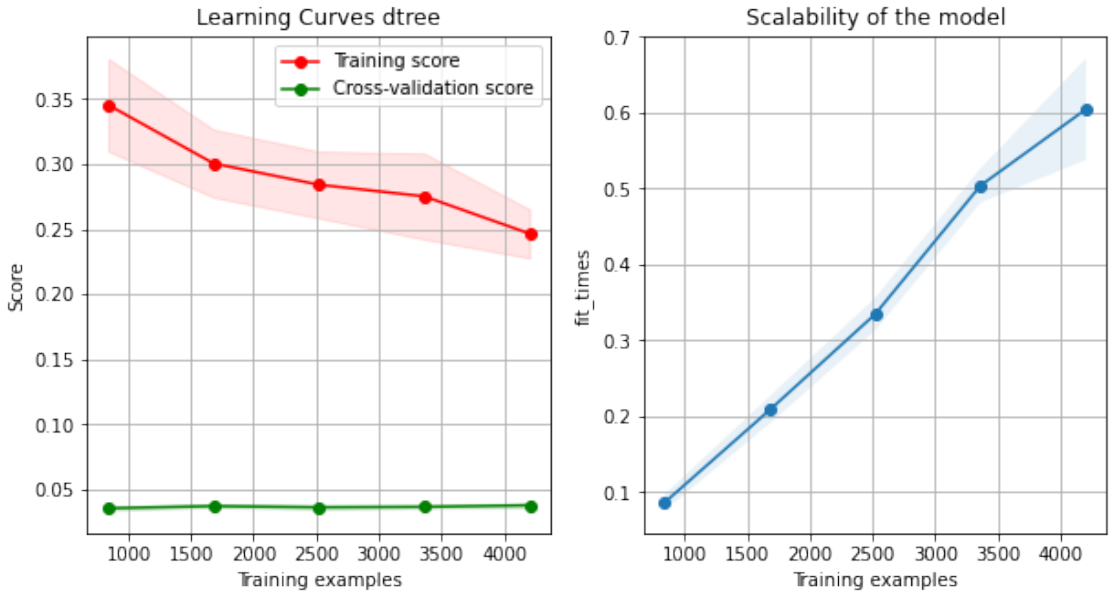
[30]:
```
['/content/drive/MyDrive/teaching_fall_2021/ml_fall_2021/HW_Assignments/HW6/save
d_models/dtree_round1_model.pkl']
```

[31]:
```
loaded_dtree_params_round1 = joblib.load(file_params_tree1)
loaded_dtree_model_round1 = joblib.load(file_model_tree1)
```

[32]:
```
plot_learning_curve(loaded_dtree_params_round1, 'Learning Curves dtree',␣
 ↪X_train, y_train, n_jobs=-1)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680:
UserWarning:

The least populated class in y has only 1 members, which is less than
n_splits=5.
```

[32]:
```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.7/dist-
packages/matplotlib/pyplot.py'>
```



[33]:
```
#let's check the train scores
print(loaded_dtree_model_round1.score(X_train,y_train))

#let's check the cross validation score
print(loaded_dtree_model_round1.best_score_)
```

```
0.23249619482496195
0.038051936775766695
```

[34]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.4,␣
↪random_state=123)#, stratify =x[['Functioning Day','Holiday']])
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor
from sklearn.preprocessing  import PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
```

[35]:
```
#cd = categorical + continuous
#print(cd)
pipeline_lin = Pipeline([

    ('one_hot_encoder',
      OneHotEncoder(variables=categorical,drop_last=True)),
```

```
    ('scalar',
      SklearnTransformerWrapper(StandardScaler(), variables = continuous)),

    ('linreg',
     LinearRegression())
])
#np.logspace(-4, -1, 10)
```

[36]:
```
np.linspace(0,1, 5)
param_grid_lin1 = {
    'scalar__transformer': [StandardScaler(), MinMaxScaler()],
    #'logreg__l1_ratio': np.linspace(0, 1, 5)
    }
grid_linreg_1 = GridSearchCV(pipeline_lin, param_grid_lin1,
                             cv=5, return_train_score= True, n_jobs=-1 )
grid_linreg_1.fit(X_train,y_train)
```

[36]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('one_hot_encoder',
                                        OneHotEncoder(drop_last=True,
                                                      variables=[])),
                                       ('scalar',
SklearnTransformerWrapper(transformer=StandardScaler(),
variables=['Hour',
'Temperature(C)',
'Humidity(%)',
'Wind '
'speed '
'(m/s)',
'Visibility '
'(10m)',
'Dew '
'point '
'temperature(C)',
'Solar '
'Radiation '
'(MJ/m2)',
'Rainfall(mm)',
'Snowfall '
'(cm)'])),
                                       ('linreg', LinearRegression())]),
             n_jobs=-1,
             param_grid={'scalar__transformer': [StandardScaler(),
                                                 MinMaxScaler()]},
             return_train_score=True)
```

[37]:
```
grid_linreg_1.best_params_
```

[37]:
```
{'scalar__transformer': StandardScaler()}
```

[38]:
```
file_params_lin1 = save_model_folder / 'linreg_round1_params.pkl'
file_model_lin1 = save_model_folder / 'linreg_round1_model.pkl'
```

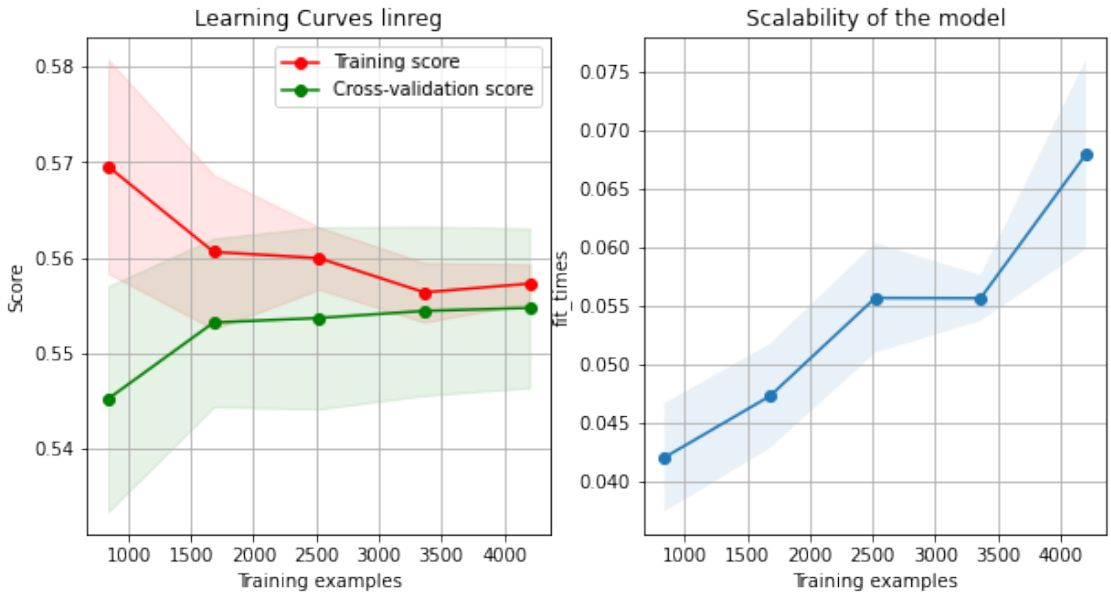
[39]:
```
joblib.dump(grid_linreg_1.best_estimator_, file_params_lin1)
joblib.dump(grid_linreg_1, file_model_lin1)
```

[39]:
```
['/content/drive/MyDrive/teaching_fall_2021/ml_fall_2021/HW_Assignments/HW6/save
d_models/linreg_round1_model.pkl']
```

[40]:
```
loaded_linreg_params_lin1 = joblib.load(file_params_lin1)
loaded_linreg_model_lin1 = joblib.load(file_model_lin1)
```

[41]:
```
plot_learning_curve(loaded_linreg_params_lin1 , 'Learning Curves linreg',␣
 ↪X_train, y_train, n_jobs=-1)
```

[41]:
```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.7/dist-
packages/matplotlib/pyplot.py'>
```



[42]:
```
#let's check the train scores
print(loaded_linreg_model_lin1.score(X_train,y_train))

#let's check the cross validation score
print(loaded_linreg_model_lin1.best_score_)
```

```
0.5570702664776213
0.5547080355909145
```

[43]:
```
print(f'Test data accauracy for lin reg : {loaded_linreg_params_lin1.
 ↪score(X_test,y_test)}')
```

```
Test data accauracy for lin reg : 0.5385365597143406
```

[44]:
```
print(f'Test data accauracy for decision tree : {loaded_dtree_params_round1.
↪score(X_test,y_test)}')
```

```
Test data accauracy for decision tree : 0.1603881278538813
```

[45]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.7/dist-packages
(0.90)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
(from xgboost) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from xgboost) (1.19.5)
```

[48]:
```
churn_pipeline = Pipeline([

('one_hot_encoder',
      OneHotEncoder(variables=categorical,drop_last=True))])

X_train=X_train.drop(['Seasons','Holiday','Functioning Day'], axis=1)
X_test = X_test.drop(['Seasons','Holiday','Functioning Day'], axis=1)
```

[49]:
```
from  xgboost import XGBRegressor
xgbc= XGBRegressor(random_state=123,early_stopping_rounds=2)
xgbc_param = {
              'max_depth' : [6],
              'n_estimators' : [100],
              'learning_rate' : [0.6],
               'min_child_weight' : [1],
                'subsample':[1]
             }
xgbc_grid = GridSearchCV(xgbc, xgbc_param,cv=5, return_train_score=True, )
xgbc_grid.fit(X_train,y_train)
```

```
[19:18:57] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:18:57] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:18:58] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:18:59] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:18:59] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:19:00] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

[49]:
```
GridSearchCV(cv=5,
             estimator=XGBRegressor(early_stopping_rounds=2, random_state=123),
             param_grid={'learning_rate': [0.6], 'max_depth': [6],
                         'min_child_weight': [1], 'n_estimators': [100],
                         'subsample': [1]},
             return_train_score=True)
```

[50]:
```
print(f'Best Mean Cross Validation Score is {xgbc_grid.best_score_}')
print(f'Best Mean Cross Validation Score is {xgbc_grid.best_params_}')
print(f'Train score is {xgbc_grid.score(X_train,y_train)}')
print(f'Test score is {xgbc_grid.score(X_test,y_test)}')
#print(f'Val score is {xgbc_grid.score(X_val,y_val)}')
```

```
Best Mean Cross Validation Score is 0.7079268452970789
Best Mean Cross Validation Score is {'learning_rate': 0.6, 'max_depth': 6,
'min_child_weight': 1, 'n_estimators': 100, 'subsample': 1}
Train score is 0.987033822783338
Test score is 0.69475228170632
```

[59]:
```
file_params_roundxg = save_model_folder / 'xg_params.pkl'
file_model_roundxg = save_model_folder / 'xg_model.pkl'
```

[60]:
```
joblib.dump(xgbc_grid.best_estimator_, file_params_roundxg)
joblib.dump(xgbc_grid, file_model_roundxg)
```

[60]:
```
['/content/drive/MyDrive/teaching_fall_2021/ml_fall_2021/HW_Assignments/HW6/save
d_models/xg_model.pkl']
```

[61]:
```
loaded_xg_params = joblib.load(file_params_roundxg)
loaded_xg_model = joblib.load(file_model_roundxg)
```

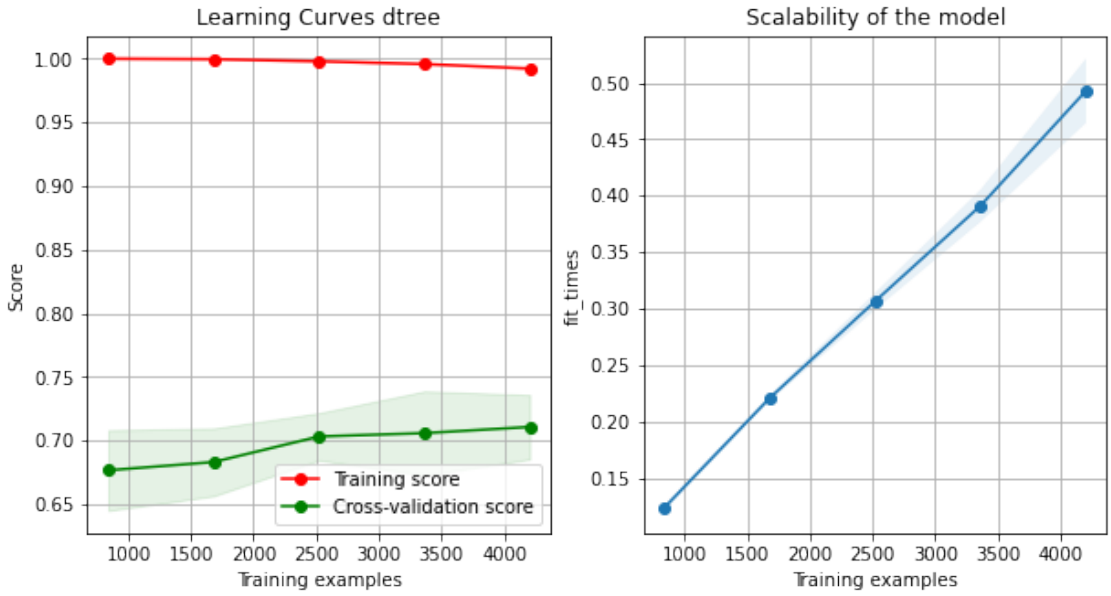
```
[19:26:12] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:12] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

[62]:
```
plot_learning_curve(loaded_xg_params, 'Learning Curves dtree', X_train, y_train)
```

```
[19:26:18] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:18] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:18] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:18] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:19] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[19:26:19] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:19] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:20] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:20] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:20] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:21] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:21] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:21] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:22] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:22] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:23] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:23] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:23] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:23] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:24] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:24] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:24] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:25] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:25] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[19:26:25] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

[62]: `<module 'matplotlib.pyplot' from '/usr/local/lib/python3.7/dist-packages/matplotlib/pyplot.py'>`



[63]:
```
#let's check the train scores
print(loaded_xg_model.score(X_train,y_train))

#let's check the cross validation score
print(loaded_xg_model.best_score_)
```

```
0.987033822783338
0.7079268452970789
```

[64]:
```
#let's check the test scores for final model
print(f'Test data accauracy for xg: {loaded_xg_model.score(X_test,y_test)}')
```

```
Test data accauracy for xg: 0.69475228170632
```

[51]:
```
# define model
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import fbeta_score
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestRegressor



model= Pipeline([

 ('rf',
       RandomForestRegressor(warm_start=True, random_state=0, oob_score=True))])
param_grid = {
```

```
    'rf__n_estimators': [200],
    'rf__max_features': ['sqrt', 'log2'],
    'rf__max_depth': np.arange(2,10),
    'rf__criterion' :['squared_error'],
    #'rf__min_samples_leaf': np.arange(2,10),
    #'rf__max_leaf_nodes': np.arange(2, 10),
}

# define evaluation procedure
#cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)


grid_randomforest= GridSearchCV(model, param_grid, cv=5, n_jobs=-1)
grid_randomforest.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names
```

[51]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('rf',
                                        RandomForestRegressor(oob_score=True,
                                                              random_state=0,
warm_start=True))]),
             n_jobs=-1,
             param_grid={'rf__criterion': ['squared_error'],
                         'rf__max_depth': array([2, 3, 4, 5, 6, 7, 8, 9]),
                         'rf__max_features': ['sqrt', 'log2'],
                         'rf__n_estimators': [200]})
```

[52]:
```
grid_randomforest.best_params_
```

[52]:
```
{'rf__criterion': 'squared_error',
 'rf__max_depth': 9,
 'rf__max_features': 'sqrt',
 'rf__n_estimators': 200}
```

[53]:
```
file_params_roundrf = save_model_folder / 'rf_round1_params.pkl'
file_model_roundrf = save_model_folder / 'rf_round1_model.pkl'
```

[54]:
```
joblib.dump(grid_randomforest.best_estimator_, file_params_roundrf)
joblib.dump(grid_randomforest, file_model_roundrf)
```

[54]:
```
['/content/drive/MyDrive/teaching_fall_2021/ml_fall_2021/HW_Assignments/HW6/save
d_models/rf_round1_model.pkl']
```

[55]:
```
loaded_rf_params = joblib.load(file_params_roundrf)
loaded_rf_model = joblib.load(file_model_roundrf)
```

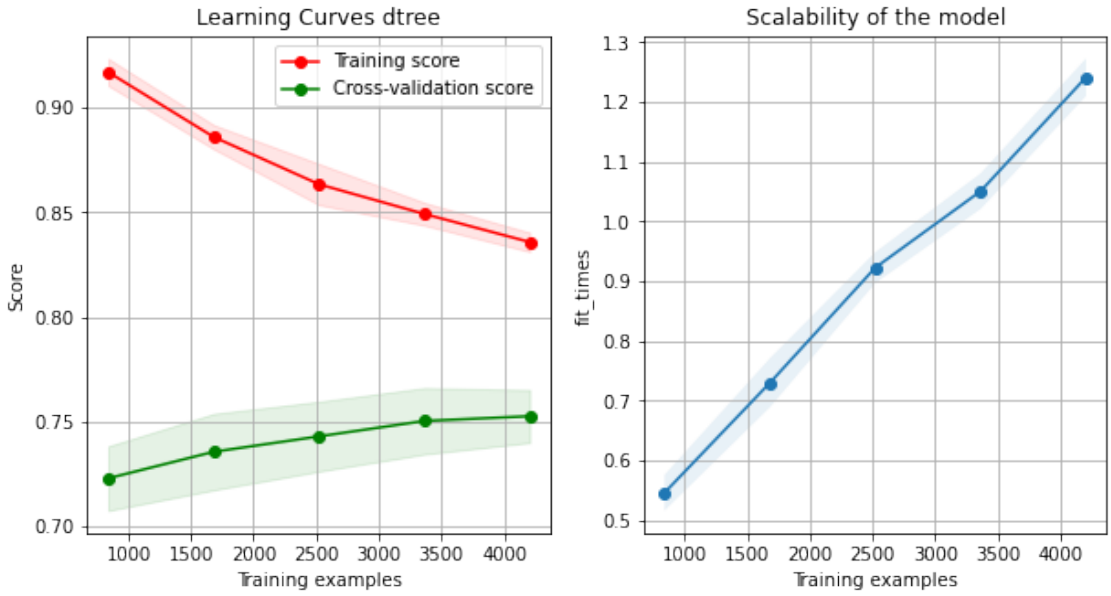
[56]:
```
plot_learning_curve(loaded_rf_params, 'Learning Curves dtree', X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
```

```
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:
```

```
X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning:

X does not have valid feature names, but RandomForestRegressor was fitted with
feature names
```

[56]:
```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.7/dist-
packages/matplotlib/pyplot.py'>
```



[57]:
```
#let's check the train scores
print(loaded_rf_model.score(X_train,y_train))

#let's check the cross validation score
print(loaded_rf_model.best_score_)
```

```
0.8270085923320966
0.752745218990871
```

[58]:
```
#let's check the test scores for final model
print(f'Test data accauracy for rf: {loaded_rf_model.score(X_test,y_test)}')
```

```
Test data accauracy for rf: 0.7312127392174269
```

Based on analysis, the learning and scaling curves, I conclude that Random Forest is the most accurate model. However, there is an overfitting issue and it does not scale well. For slightly less overfitting but a more scalable model, xgboost might be more effective.

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('Sameeullah_File1_HW6.ipynb')
```

```
--2021-12-21 19:28:52--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443... connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: colab_pdf.py

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2021-12-21 19:28:52 (23.7 MB/s) - colab_pdf.py saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```