Body Fat Prediction Dataset

October 16, 2023

1 Project: Body Fat Prediction Dataset

Body fat estimates and various body circumference measurements for 252 men.

For more detail, check Kaggle.

1.1 Index

The project is split into five parts:

Part 1: We treated the data removing the outliers and using the filter method to choose only the most important features. We used Linear regression and RF to make predictions with the new data set. Ultimately, we trained an XGB regressor on the complete data set and compared the results.

Part 2: We used the auto-Sklern library to do algorithm selection and hyperparameter tuning.

Part 3: We trained two different Neural Networks.

Part 4: We used Auto-PyTorch to optimize the network architecture and the training hyperparameters.

Part 5: We used PyTorch Tabular and trained the following models: Category Embedding, GATE, NODE, and TabNet.

1.2 About Dataset

1.2.1 Context

Lists estimates of the percentage of body fat determined by underwater weighing and various body circumference measurements for 252 men.

1.2.2 Educational use of the dataset

This data set can be used to illustrate multiple regression techniques. Accurate measurement of body fat is inconvenient/costly and it is desirable to have easy methods of estimating body fat that are not inconvenient/costly.

1.3 Analizing Feature importance

We start by analyzing the data and the correlation between the features. The goal is to identify outliers and remove unecessary features.

[ ]:
```python
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sn
from sklearn.metrics import r2_score
sn.set(style='whitegrid')
```

[ ]:
```python
bf = pd.read_csv('bodyfat.csv')
bf.head()
```

[ ]:
```
   Density  BodyFat  Age  Weight  Height  Neck  Chest  Abdomen    Hip  Thigh  \
0   1.0708     12.3   23  154.25   67.75  36.2   93.1     85.2   94.5   59.0
1   1.0853      6.1   22  173.25   72.25  38.5   93.6     83.0   98.7   58.7
2   1.0414     25.3   22  154.00   66.25  34.0   95.8     87.9   99.2   59.6
3   1.0751     10.4   26  184.75   72.25  37.4  101.8     86.4  101.2   60.1
4   1.0340     28.7   24  184.25   71.25  34.4   97.3    100.0  101.9   63.2

   Knee  Ankle  Biceps  Forearm  Wrist
0  37.3   21.9    32.0     27.4   17.1
1  37.3   23.4    30.5     28.9   18.2
2  38.9   24.0    28.8     25.2   16.6
3  37.3   22.8    32.4     29.4   18.2
4  42.2   24.0    32.2     27.7   17.7
```

2 Outliers

[ ]:
```python
for feature in bf.columns:
    data = bf[feature]
    Q3, Q1 = data.quantile(0.75), data.quantile(0.25)
    IQR = Q3 - Q1

    outliers = ((data < (Q1 - 1.5 * IQR)) | (data > (Q3 + 1.5 * IQR)))
    print(feature)
    print(data[outliers])
```

```
Density
215    0.995
Name: Density, dtype: float64
BodyFat
215    47.5
Name: BodyFat, dtype: float64
Age
Series([], Name: Age, dtype: int64)
Weight
38    363.15
```

```
40    262.75
Name: Weight, dtype: float64
Height
41    29.5
Name: Height, dtype: float64
Neck
38     51.2
44     31.5
105    31.1
Name: Neck, dtype: float64
Chest
38    136.2
40    128.3
Name: Chest, dtype: float64
Abdomen
38     148.1
40     126.2
215    122.1
Name: Abdomen, dtype: float64
Hip
34    116.1
38    147.7
40    125.6
Name: Hip, dtype: float64
Thigh
38     87.3
40     72.5
151    72.9
168    74.4
Name: Thigh, dtype: float64
Knee
38     49.1
191    45.0
243    46.0
Name: Knee, dtype: float64
Ankle
30    33.9
38    29.6
85    33.7
Name: Ankle, dtype: float64
Biceps
38    45.0
Name: Biceps, dtype: float64
Forearm
44     23.1
158    34.9
174    21.0
205    23.1
```

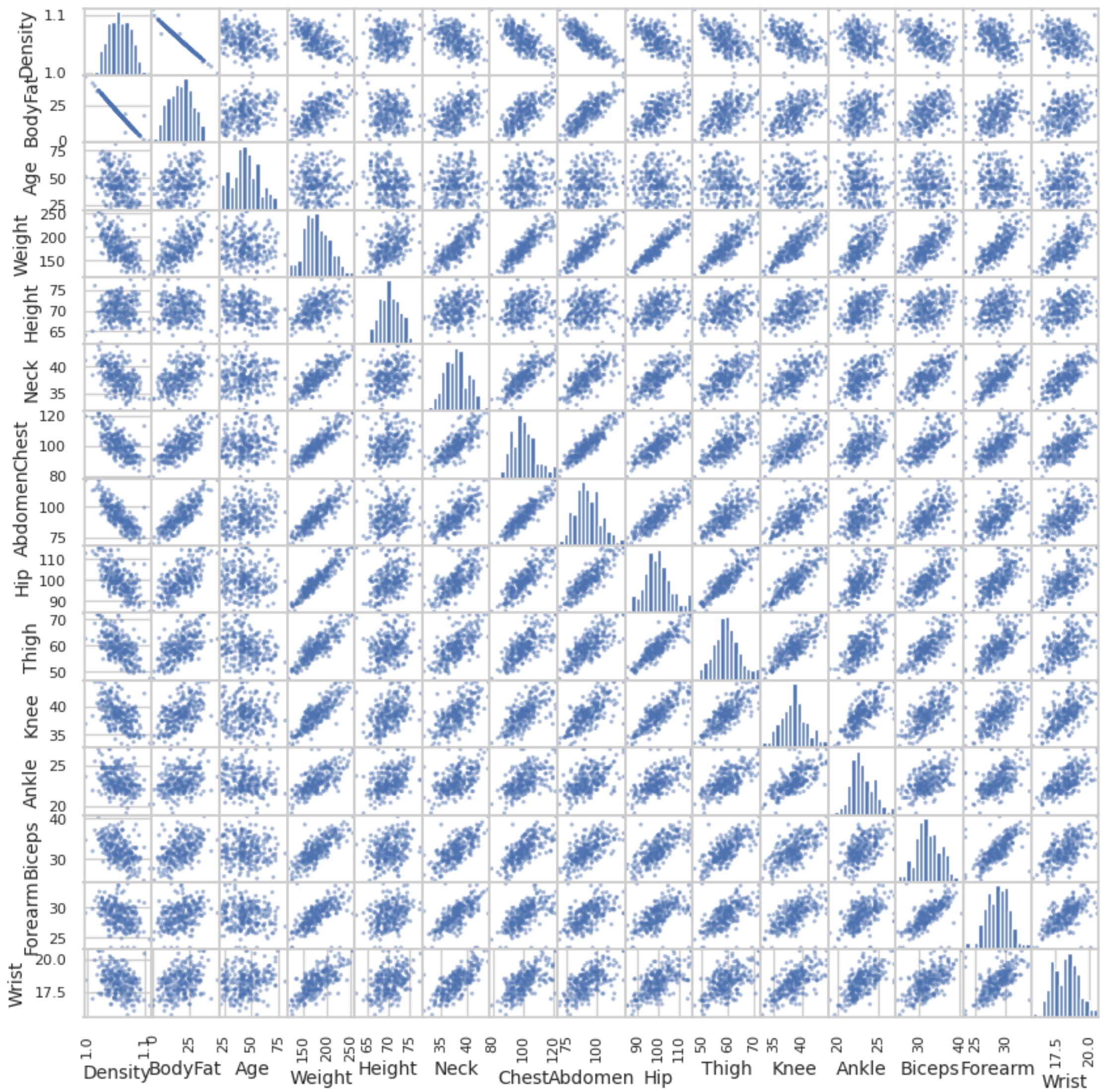
```
225    22.0
Name: Forearm, dtype: float64
Wrist
38     21.4
40     21.4
225    15.8
251    20.9
Name: Wrist, dtype: float64
```

Notice that we have a considerable number of outliears. Since out data set is small, we cannot afford trimming the outliers. So we will fix the values by capping.

```
[ ]: for feature in bf.columns:
         data = bf[feature]
         Q3, Q1 = data.quantile(0.75), data.quantile(0.25)
         IQR = Q3 - Q1
         upper_lim = (Q3 + 1.5 * IQR)
         lower_lim = (Q1 - 1.5 * IQR)
         bf[feature] = np.where( bf[feature] > upper_lim, upper_lim, bf[feature])
         bf[feature] = np.where( bf[feature] < lower_lim, lower_lim, bf[feature])
```

2.0.1 Visualize the data using scatter plot

```
[ ]: scatter = pd.plotting.scatter_matrix(bf, marker = '.', s=20, hist_kwds={'bins':
      ↪15}, figsize=(10,10))
     #y labels
     temp1=[plt.setp(item.yaxis.get_label(), 'size', 10) for item in scatter.ravel()]
     #x labels
     temp2=[plt.setp(item.xaxis.get_label(), 'size', 10) for item in scatter.ravel()]
```



Notice that some features (like age, for instance), have a small correlation with body fat. This means that these features won't contribute to determine out target.

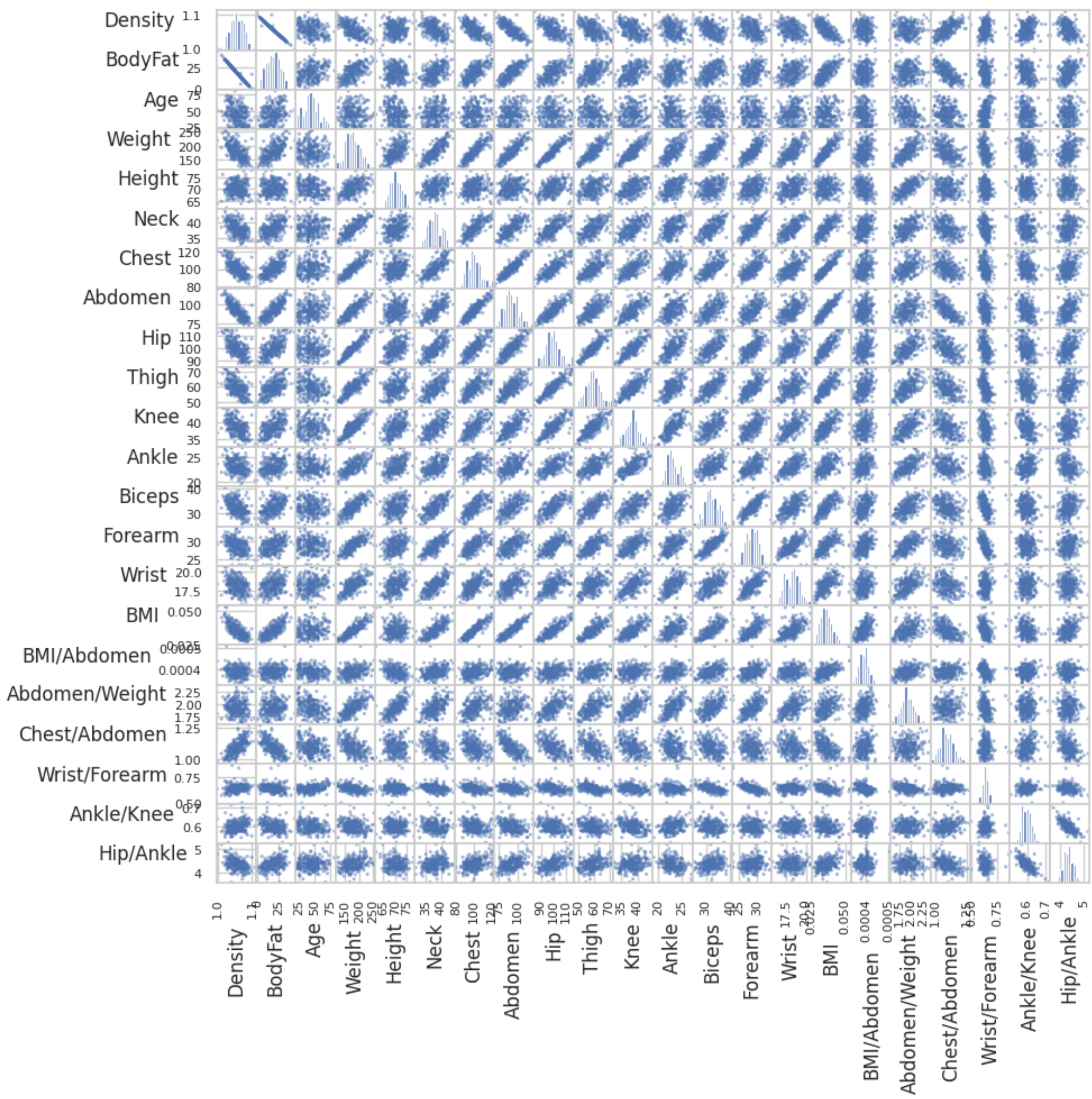
Now, we will create new features by combining the ones we already have (feature engineering).

2.0.2 Feature engineering

```
bf['BMI'] = bf['Weight']/(bf['Height']*bf['Height'])

bf['BMI/Abdomen'] = bf['BMI']/bf['Abdomen']

bf['Abdomen/Weight'] = bf['Weight']/bf['Abdomen']

bf['Chest/Abdomen'] = bf['Chest']/bf['Abdomen']
```

```python
bf['Wrist/Forearm'] = bf['Wrist']/bf['Forearm']

bf['Ankle/Knee'] = bf['Ankle']/bf['Knee']

bf['Hip/Ankle'] = bf['Hip']/bf['Ankle']
```

[ ]:
```python
scatter = pd.plotting.scatter_matrix(bf, marker = '.', s=20, hist_kwds={'bins':
 ↪15}, figsize=(10,10))
for ax in scatter.flatten():
    ax.xaxis.label.set_rotation(90)
    ax.yaxis.label.set_rotation(0)
    ax.yaxis.label.set_ha('right')

plt.tight_layout()
plt.gcf().subplots_adjust(wspace=0, hspace=0)
plt.show()
```



We will use the Filter method to select the relavant features to our models.

Saving the full data for latter.

[ ]:
```
Y0 = bf['BodyFat']
X0 = bf.iloc[:,2:]
```

[ ]:
```
X0_train, X0_test, Y0_train, Y0_test = train_test_split(X0, Y0, test_size=0.2, random_state=0)
```

[ ]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(X0_train)
```

```
#normalize the features in the training set
X0_train_s = scaler.transform(X0_train)
#normalize the features in the test set
X0_test_s = scaler.transform(X0_test)
#normalize the features in the validation set
#X_val_s = scaler.transform(X_val)
```

[ ]:
```
X0.to_csv('X.csv', index=False)
Y0.to_csv('Y.csv', index=False)
```

3 Filter

1) Identify input features having high correlation with target variable.

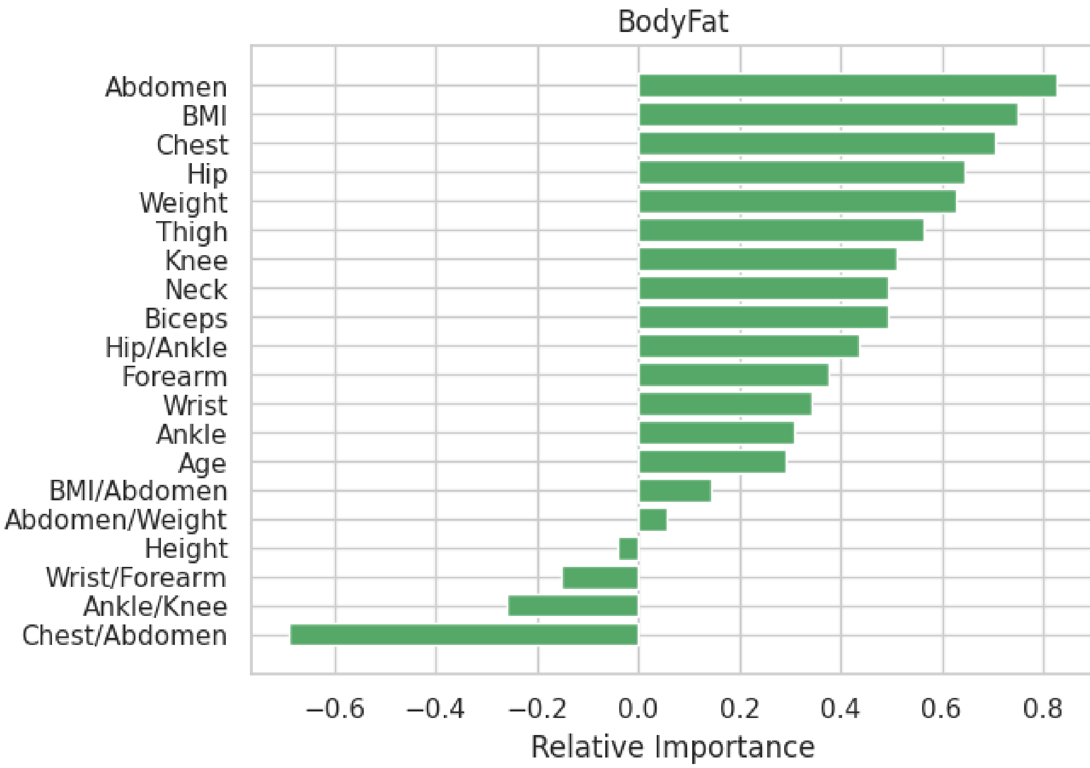
[ ]:
```
bf.drop('Density', axis=1, inplace=True)
```

[ ]:
```
importances = bf.drop('BodyFat', axis=1).apply(lambda x: x.corr(bf.BodyFat))
indices = np.argsort(importances)
print(importances[indices])
```

```
Chest/Abdomen    -0.687958
Ankle/Knee       -0.257773
Wrist/Forearm    -0.152760
Height           -0.039527
Abdomen/Weight    0.055757
BMI/Abdomen       0.143452
Age               0.292100
Ankle             0.307900
Wrist             0.341262
Forearm           0.375617
Hip/Ankle         0.437836
Biceps            0.493216
Neck              0.493320
Knee              0.509030
Thigh             0.565269
Weight            0.627178
Hip               0.645717
Chest             0.706564
BMI               0.749969
Abdomen           0.827888
dtype: float64
```

[ ]:
```
names=list(bf.drop('BodyFat', axis=1).columns)
plt.title('BodyFat')
plt.barh(range(len(indices)), importances[indices], color='g', align='center')
plt.yticks(range(len(indices)), [names[i] for i in indices])
plt.xlabel('Relative Importance')
```

```
plt.show()
```



We set the threshold to the absolute value of 0.35. We keep input features only if the correlation of the input feature with the target variable is greater than 0.35

[ ]:
```
selected_features = []
for i in range(0, len(indices)):
    if np.abs(importances[i])>0.3:
        selected_features.append(names[i])
        print(names[i])
```

```
Weight
Neck
Chest
Abdomen
Hip
Thigh
Knee
Ankle
Biceps
Forearm
Wrist
BMI
```

```
Chest/Abdomen
Hip/Ankle
```

[ ]:
```
bf = bf[['BodyFat']+selected_features]
bf.head()
```

[ ]:

	BodyFat	Weight	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	\
0	12.3	154.25	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	
1	6.1	173.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	
2	25.3	154.00	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	
3	10.4	184.75	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	
4	28.7	184.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	

	Forearm	Wrist	BMI	Chest/Abdomen	Hip/Ankle
0	27.4	17.1	0.033605	1.092723	4.315068
1	28.9	18.2	0.033189	1.127711	4.217949
2	25.2	16.6	0.035087	1.089875	4.133333
3	29.4	18.2	0.035392	1.178241	4.438596
4	27.7	17.7	0.036294	0.973000	4.245833

2) Identify input features that have a low correlation with other independent variables.

[ ]:
```
for i in range(0,len(bf.columns)):
    for j in  range(0,len(bf.columns)):
        if i!=j:
            corr_1=np.abs(bf[bf.columns[i]].corr(bf[bf.columns[j]]))
            if corr_1 <0.3:
                print( bf.columns[i] , " is not correlated  with ", bf.
 ↪columns[j])
            elif corr_1>0.75:
                print( bf.columns[i] , " is highly  correlated  with ", bf.
 ↪columns[j])
```

```
BodyFat  is highly  correlated  with  Abdomen
Weight  is highly  correlated  with  Neck
Weight  is highly  correlated  with  Chest
Weight  is highly  correlated  with  Abdomen
Weight  is highly  correlated  with  Hip
Weight  is highly  correlated  with  Thigh
Weight  is highly  correlated  with  Knee
Weight  is highly  correlated  with  Biceps
Weight  is highly  correlated  with  BMI
Neck  is highly  correlated  with  Weight
Neck  is highly  correlated  with  Chest
Neck  is not correlated  with  Hip/Ankle
Chest  is highly  correlated  with  Weight
Chest  is highly  correlated  with  Neck
Chest  is highly  correlated  with  Abdomen
```

```
Chest  is highly  correlated  with  Hip
Chest  is highly  correlated  with  BMI
Abdomen  is highly  correlated  with  BodyFat
Abdomen  is highly  correlated  with  Weight
Abdomen  is highly  correlated  with  Chest
Abdomen  is highly  correlated  with  Hip
Abdomen  is highly  correlated  with  BMI
Hip  is highly  correlated  with  Weight
Hip  is highly  correlated  with  Chest
Hip  is highly  correlated  with  Abdomen
Hip  is highly  correlated  with  Thigh
Hip  is highly  correlated  with  Knee
Hip  is highly  correlated  with  BMI
Thigh  is highly  correlated  with  Weight
Thigh  is highly  correlated  with  Hip
Thigh  is highly  correlated  with  Knee
Thigh  is highly  correlated  with  BMI
Knee  is highly  correlated  with  Weight
Knee  is highly  correlated  with  Hip
Knee  is highly  correlated  with  Thigh
Knee  is not correlated  with  Hip/Ankle
Ankle  is not correlated  with  Chest/Abdomen
Biceps  is highly  correlated  with  Weight
Biceps  is not correlated  with  Hip/Ankle
Forearm  is not correlated  with  Chest/Abdomen
Forearm  is not correlated  with  Hip/Ankle
Wrist  is not correlated  with  Chest/Abdomen
Wrist  is not correlated  with  Hip/Ankle
BMI  is highly  correlated  with  Weight
BMI  is highly  correlated  with  Chest
BMI  is highly  correlated  with  Abdomen
BMI  is highly  correlated  with  Hip
BMI  is highly  correlated  with  Thigh
Chest/Abdomen  is not correlated  with  Ankle
Chest/Abdomen  is not correlated  with  Forearm
Chest/Abdomen  is not correlated  with  Wrist
Hip/Ankle  is not correlated  with  Neck
Hip/Ankle  is not correlated  with  Knee
Hip/Ankle  is not correlated  with  Biceps
Hip/Ankle  is not correlated  with  Forearm
Hip/Ankle  is not correlated  with  Wrist
```

[ ]:
```
uncorrelated_features = ['Abdomen', 'Neck', 'Hip/Ankle', 'Knee', 'Ankle',
 ↪'Biceps', 'Chest/Abdomen', 'Forearm', 'Wrist']
bf = bf[['BodyFat']+uncorrelated_features]
bf.head()
```

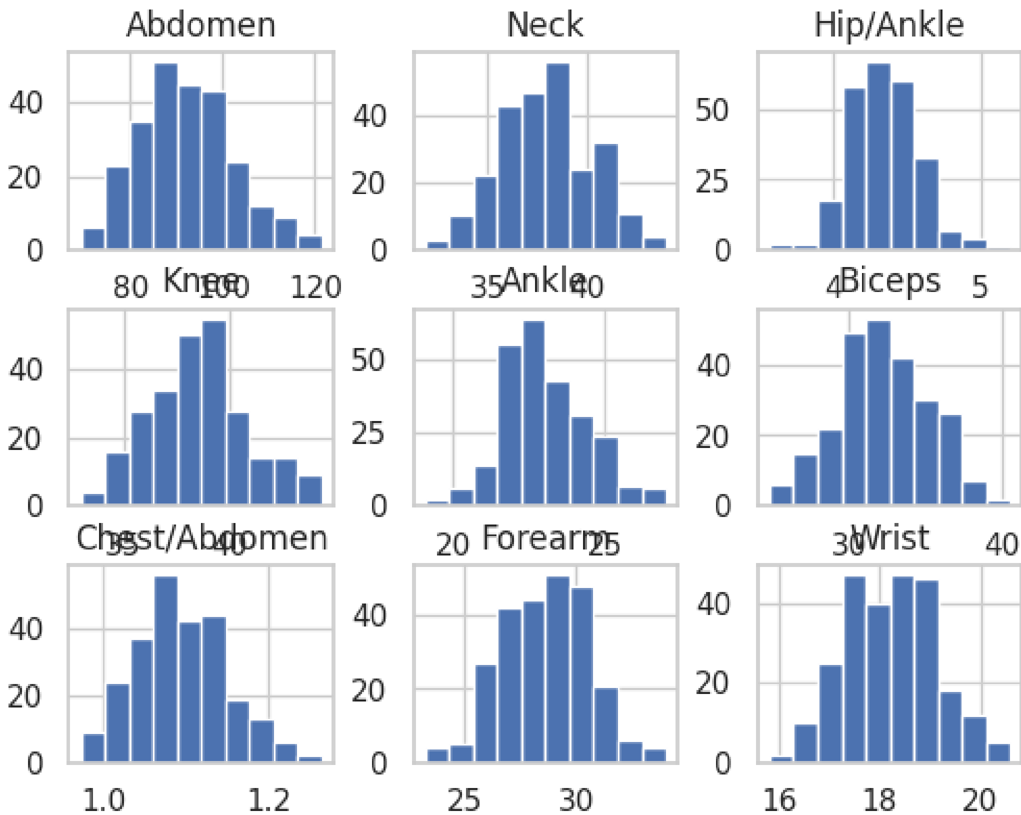
```
[ ]:    BodyFat  Abdomen  Neck  Hip/Ankle  Knee  Ankle  Biceps  Chest/Abdomen  \
     0     12.3     85.2  36.2   4.315068  37.3   21.9    32.0       1.092723
     1      6.1     83.0  38.5   4.217949  37.3   23.4    30.5       1.127711
     2     25.3     87.9  34.0   4.133333  38.9   24.0    28.8       1.089875
     3     10.4     86.4  37.4   4.438596  37.3   22.8    32.4       1.178241
     4     28.7    100.0  34.4   4.245833  42.2   24.0    32.2       0.973000

        Forearm  Wrist
     0     27.4   17.1
     1     28.9   18.2
     2     25.2   16.6
     3     29.4   18.2
     4     27.7   17.7
```

3.0.1 Split the data into a Training Set and a Testing/Test Set and normalization/standardization

```
[ ]: Y = bf['BodyFat']
     X = bf.iloc[:,1:]
```

```
[ ]: X.hist()
```

```
[ ]: array([[<Axes: title={'center': 'Abdomen'}>,
             <Axes: title={'center': 'Neck'}>,
             <Axes: title={'center': 'Hip/Ankle'}>],
            [<Axes: title={'center': 'Knee'}>,
             <Axes: title={'center': 'Ankle'}>,
             <Axes: title={'center': 'Biceps'}>],
            [<Axes: title={'center': 'Chest/Abdomen'}>,
             <Axes: title={'center': 'Forearm'}>,
             <Axes: title={'center': 'Wrist'}>]], dtype=object)
```



Notice that the data seems to have a gaussian distribuiton. Hence, we will use StandardScaler

```
[ ]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
     #X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size=0.1, random_state=0)
```

```
[ ]: X.to_csv('X.csv', index=False)
     Y.to_csv('Y.csv', index=False)
```

```
[ ]: #convert pandas dataframe to numpy array

     X_train=X_train.values

     Y_train=Y_train.values

     X_test=X_test.values

     Y_test=Y_test.values

     #X_val=X_val.values
```

```
#Y_val=Y_val.values
```

Feature standardization

[ ]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X_train)

#normalize the features in the training set
X_train_s = scaler.transform(X_train)
#normalize the features in the test set
X_test_s = scaler.transform(X_test)
#normalize the features in the validation set
#X_val_s = scaler.transform(X_val)
```

3.0.2 Using RF Model and analizing feature importance

[ ]:
```
def my_scorer(model, X, y):
    y_pred = model.predict(X)
    #MSE = np.mean((y_pred - y)**2)3
    #MAE = np.mean(np.abs(y_pred - y))
    MAPE =  np.mean(np.abs(y_pred - y)/y)
    return -MAPE
```

[ ]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

params = [{'max_depth': np.arange(10, 150, 5),
        'n_estimators': np.arange(64, 125, 10)
          }]


RF = GridSearchCV(RandomForestRegressor(),
                      param_grid=params,
                      scoring=my_scorer,
                      cv=3)
RF.fit(X_train, Y_train)
RF.best_params_
```

[ ]:
```
{'max_depth': 75, 'n_estimators': 64}
```

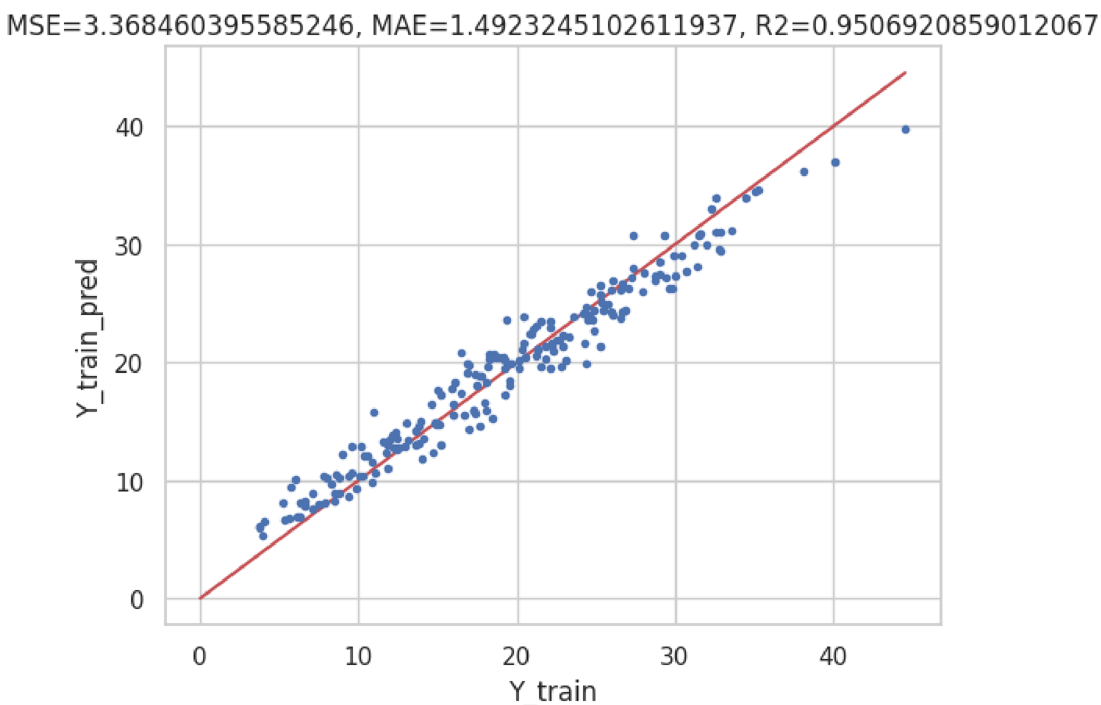
[ ]:
```
model_best=RF.best_estimator_

model_best.fit(X_train, Y_train)
Y_train_pred = model_best.predict(X_train)
```

```
Y_test_pred = model_best.predict(X_test)
```

Trainig results:

```
[ ]: MSE = np.mean((Y_train - Y_train_pred)**2)
     MAE = np.mean(np.abs(Y_train - Y_train_pred))
     R2 = r2_score(Y_train, Y_train_pred)
     #
     ymax=np.max([Y_train.max(), Y_train_pred.max()])
     plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
     plt.plot(Y_train, Y_train_pred, '.')
     plt.xlabel('Y_train')
     plt.ylabel('Y_train_pred')
     plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

```
[ ]: Text(0.5, 1.0, 'MSE=3.368460395585246, MAE=1.4923245102611937,
     R2=0.9506920859012067')
```

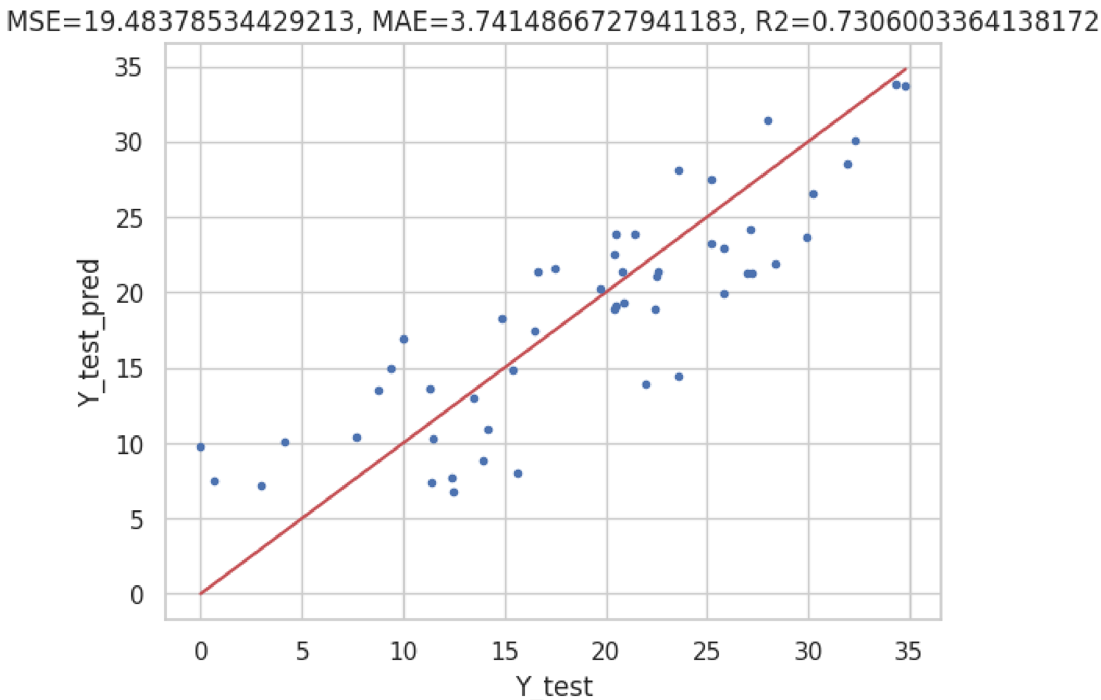


Testing results:

```
[ ]: MSE = np.mean((Y_test - Y_test_pred)**2)
     MAE = np.mean(np.abs(Y_test - Y_test_pred))
     R2 = r2_score(Y_test, Y_test_pred)
     #
```

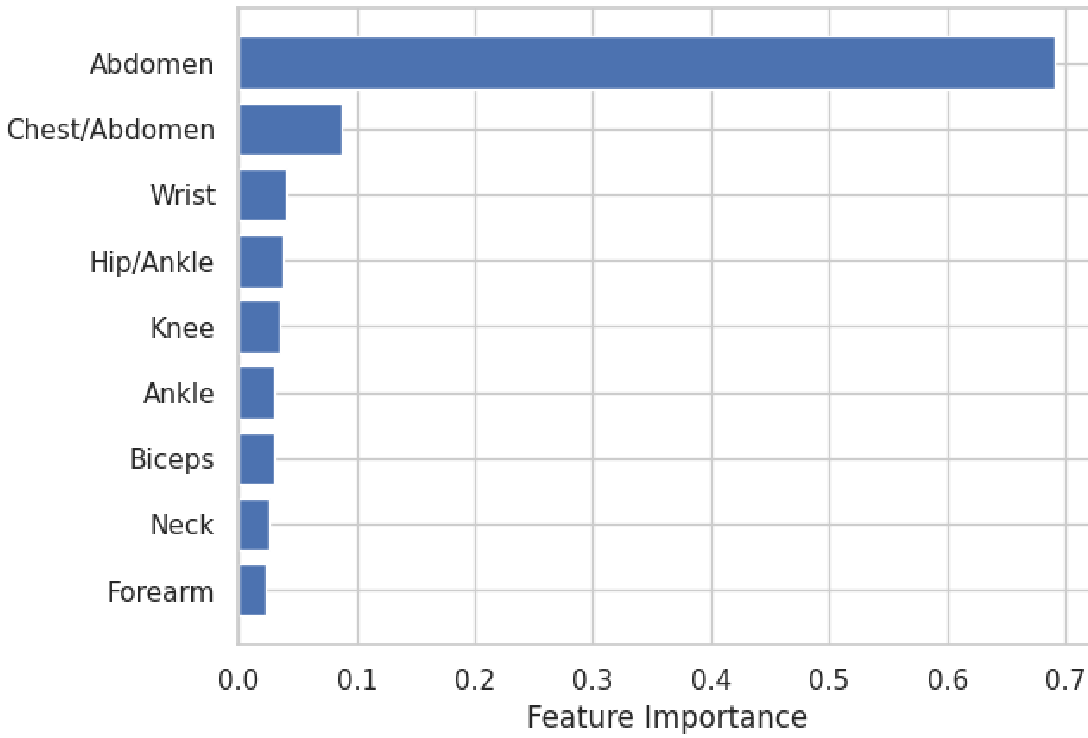
```
ymax=np.max([Y_test.max(), Y_test_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_test, Y_test_pred, '.')
plt.xlabel('Y_test')
plt.ylabel('Y_test_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[ ]: Text(0.5, 1.0, 'MSE=19.48378534429213, MAE=3.7414866727941183, R2=0.7306003364138172')



[ ]:
```
sort = model_best.feature_importances_.argsort()
plt.barh(X.columns[sort], model_best.feature_importances_[sort])
plt.xlabel("Feature Importance")
```

[ ]: Text(0.5, 0, 'Feature Importance')



3.1 Linear Regression

[ ]:
```
from sklearn.linear_model import LinearRegression
model_LN = LinearRegression()
model_LN.fit(X_train_s, Y_train)
```

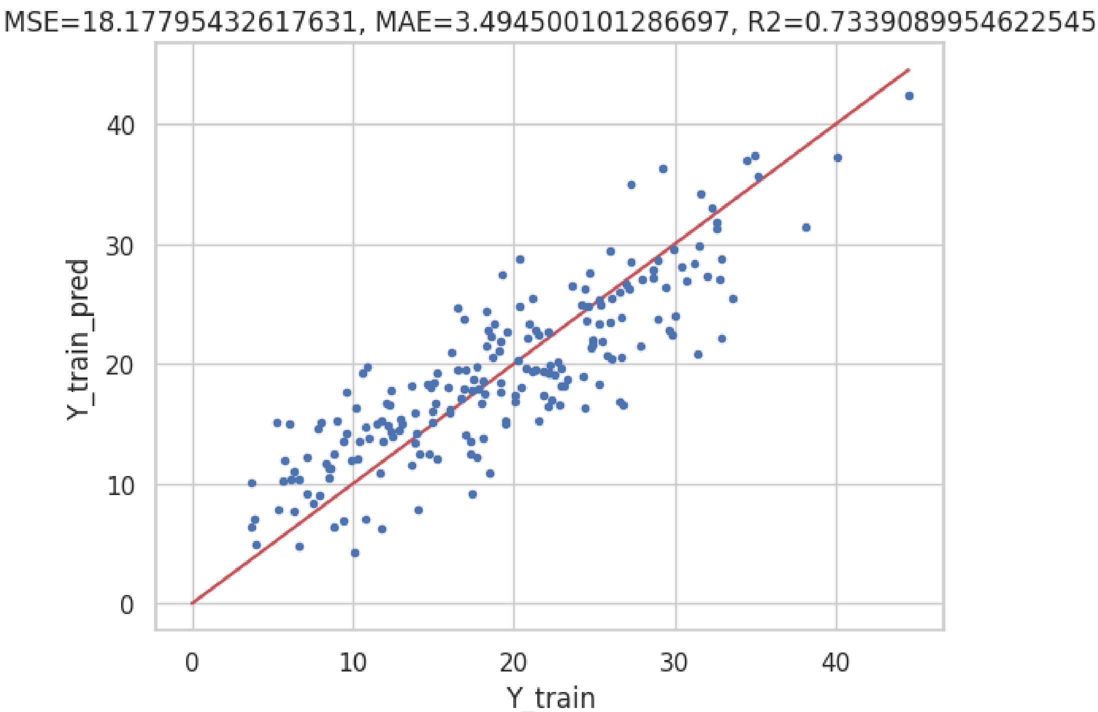
[ ]:
```
LinearRegression()
```

[ ]:
```
Y_train_pred=model_LN.predict(X_train_s)
Y_test_pred=model_LN.predict(X_test_s)
```

Trainig results:

[ ]:
```
MSE = np.mean((Y_train - Y_train_pred)**2)
MAE = np.mean(np.abs(Y_train - Y_train_pred))
R2 = r2_score(Y_train, Y_train_pred)
#
ymax=np.max([Y_train.max(), Y_train_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_train, Y_train_pred, '.')
plt.xlabel('Y_train')
plt.ylabel('Y_train_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

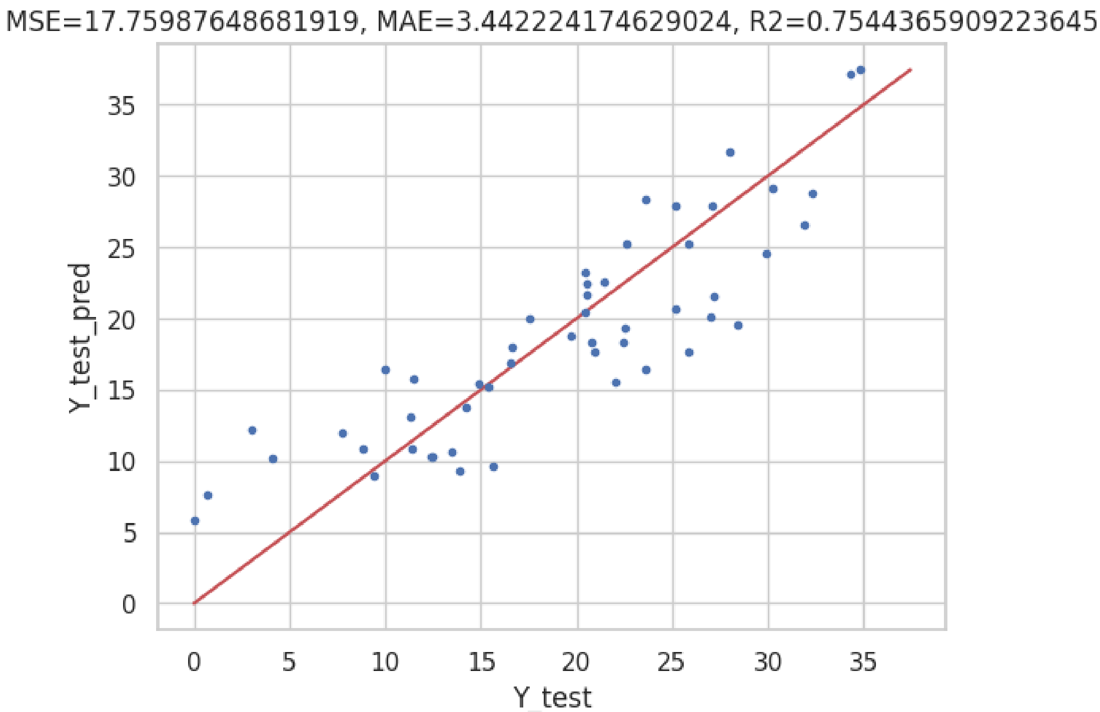
[ ]: Text(0.5, 1.0, 'MSE=18.17795432617631, MAE=3.494500101286697, R2=0.7339089954622545')



Testing results:

[ ]:
```
MSE = np.mean((Y_test - Y_test_pred)**2)
MAE = np.mean(np.abs(Y_test - Y_test_pred))
R2 = r2_score(Y_test, Y_test_pred)
#
ymax=np.max([Y_test.max(), Y_test_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_test, Y_test_pred, '.')
plt.xlabel('Y_test')
plt.ylabel('Y_test_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[ ]: Text(0.5, 1.0, 'MSE=17.75987648681919, MAE=3.442224174629024, R2=0.7544365909223645')



3.2 XGBRegressor

We will star running the XGB Regressor on the filtered data, and then we will run on the full data set to see if it performs differently.

3.2.1 Filtered Dataset:

[ ]:
```
from xgboost.sklearn import XGBRegressor
from sklearn.model_selection import GridSearchCV
#try to use GridSearchCV to do 10-fold cross-valiation to find the best␣
 ↪max_depth
#set random_state=0, objective='reg:squarederror'
#read the comments in the cell of "my_scorer", or you can use the "built-in"␣
 ↪scorer in sk-learn

params = [{'max_depth': np.arange(1, 150, 5),
           'n_estimators': np.arange(30, 125, 10)}]

gs_XGB = GridSearchCV(XGBRegressor(),
                      param_grid=params,
                      scoring=my_scorer,
                      cv=3)
```

```
gs_XGB.fit(X_train_s, Y_train)
gs_XGB.best_params_
```

[ ]: {'max_depth': 1, 'n_estimators': 40}

[ ]:
```
model_best=gs_XGB.best_estimator_
model_best
```

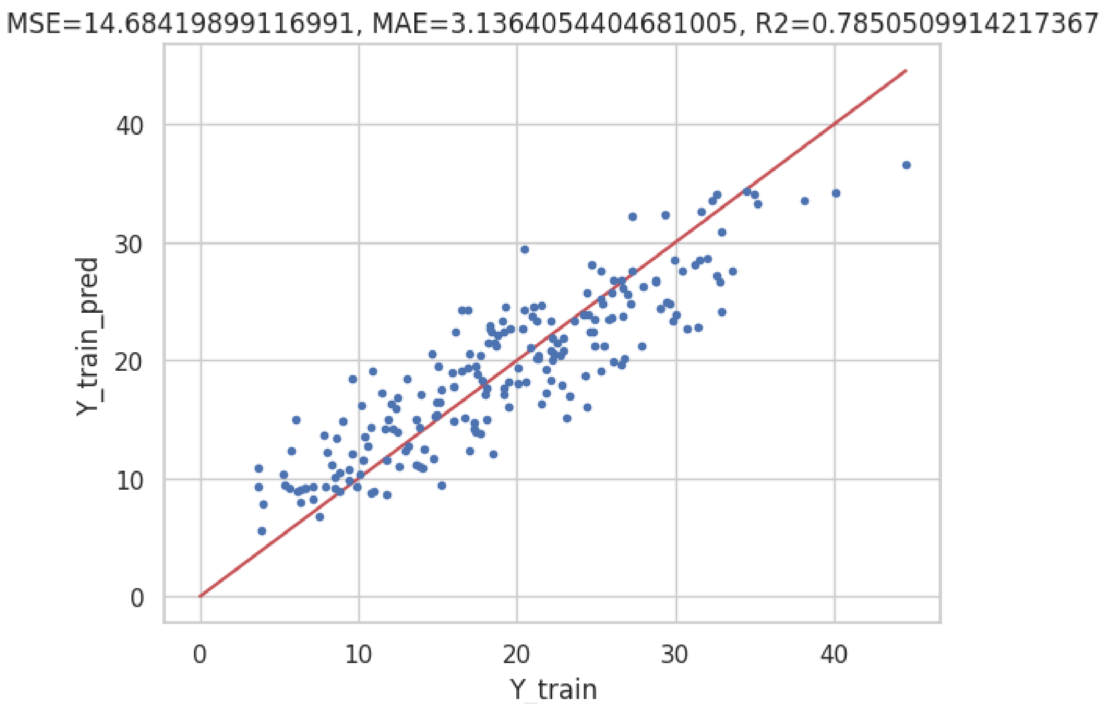
[ ]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=1, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=40, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, …)
```

[ ]:
```
model_best.fit(X_train_s, Y_train)
Y_train_pred = model_best.predict(X_train_s)
Y_test_pred = model_best.predict(X_test_s)
```

Trainig results:

[ ]:
```
MSE = np.mean((Y_train - Y_train_pred)**2)
MAE = np.mean(np.abs(Y_train - Y_train_pred))
R2 = r2_score(Y_train, Y_train_pred)
#
ymax=np.max([Y_train.max(), Y_train_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_train, Y_train_pred, '.')
plt.xlabel('Y_train')
plt.ylabel('Y_train_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

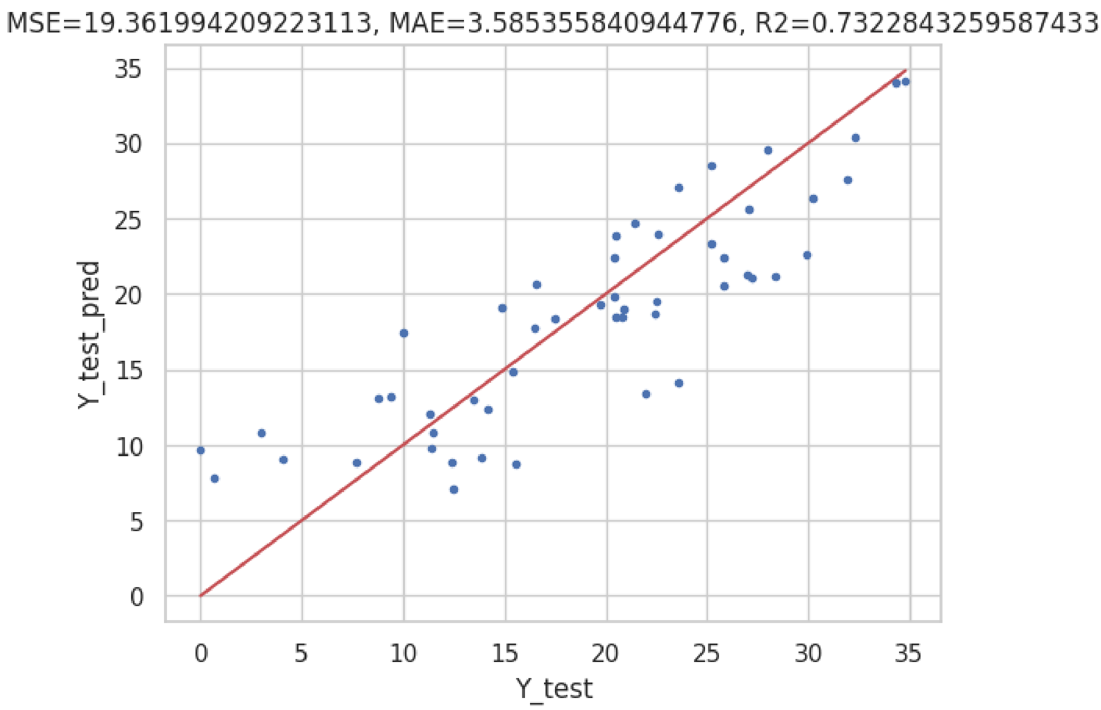
[ ]:
```
Text(0.5, 1.0, 'MSE=14.68419899116991, MAE=3.1364054404681005,
R2=0.7850509914217367')
```



Testing results:

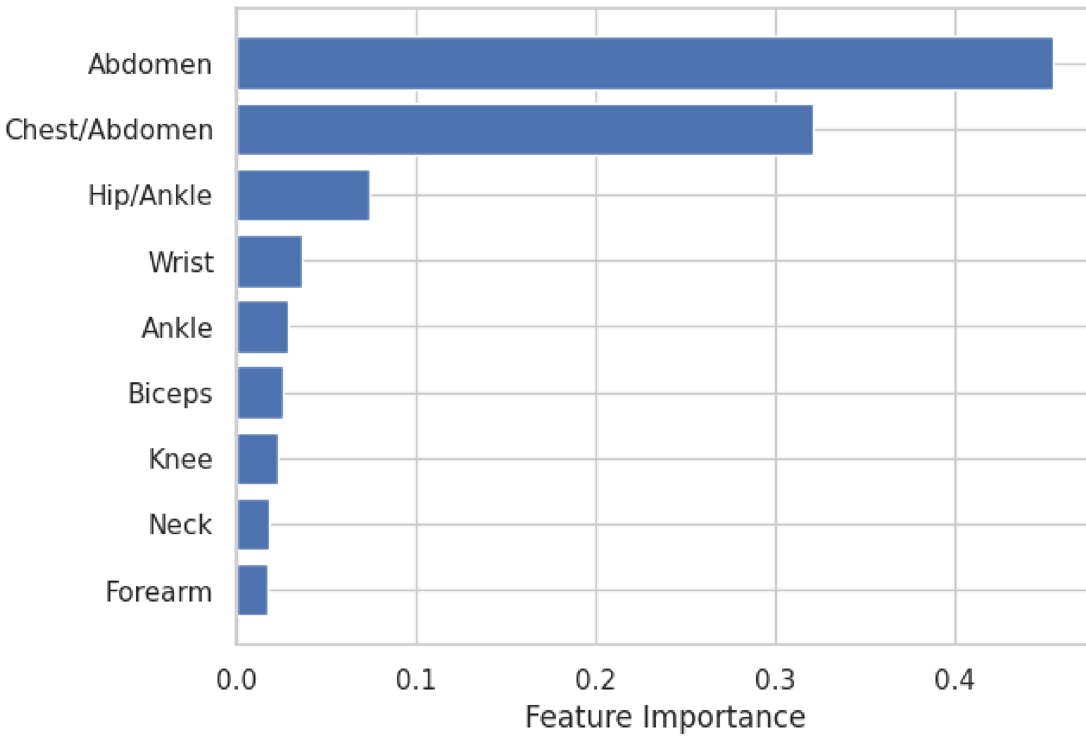
[ ]:
```
MSE = np.mean((Y_test - Y_test_pred)**2)
MAE = np.mean(np.abs(Y_test - Y_test_pred))
R2 = r2_score(Y_test, Y_test_pred)
#
ymax=np.max([Y_test.max(), Y_test_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_test, Y_test_pred, '.')
plt.xlabel('Y_test')
plt.ylabel('Y_test_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[ ]:
```
Text(0.5, 1.0, 'MSE=19.361994209223113, MAE=3.585355840944776,
R2=0.7322843259587433')
```



[ ]:
```
sort = model_best.feature_importances_.argsort()
plt.barh(X.columns[sort], model_best.feature_importances_[sort])
plt.xlabel("Feature Importance")
```

[ ]:
```
Text(0.5, 0, 'Feature Importance')
```



3.2.2 Full Dataset:

[ ]:
```
gs_XGB.fit(X0_train_s, Y0_train)
gs_XGB.best_params_
```

[ ]:
```
{'max_depth': 1, 'n_estimators': 100}
```

[ ]:
```
model_best=gs_XGB.best_estimator_
model_best
```

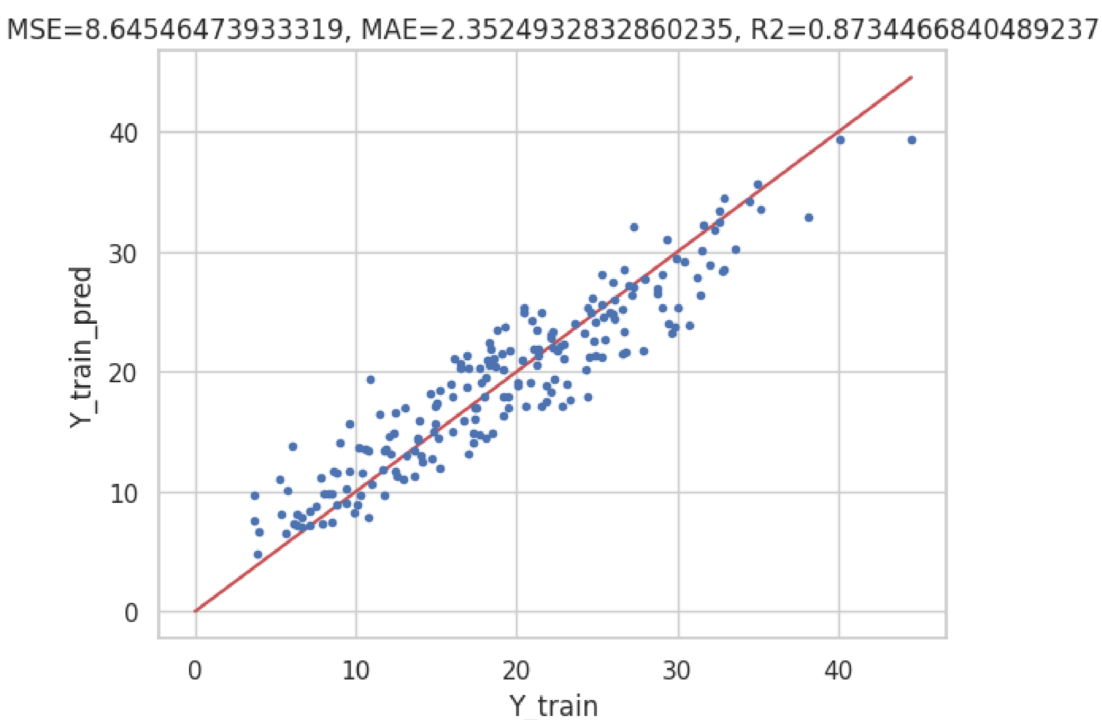
[ ]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=1, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, …)
```

[ ]:
```
model_best.fit(X0_train_s, Y0_train)
Y0_train_pred = model_best.predict(X0_train_s)
Y0_test_pred = model_best.predict(X0_test_s)
```

Trainig results:

[ ]:
```
MSE = np.mean((Y0_train - Y0_train_pred)**2)
MAE = np.mean(np.abs(Y0_train - Y0_train_pred))
R2 = r2_score(Y0_train, Y0_train_pred)
#
ymax=np.max([Y0_train.max(), Y0_train_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y0_train, Y0_train_pred, '.')
plt.xlabel('Y_train')
plt.ylabel('Y_train_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[ ]:
```
Text(0.5, 1.0, 'MSE=8.64546473933319, MAE=2.3524932832860235,
R2=0.8734466840489237')
```

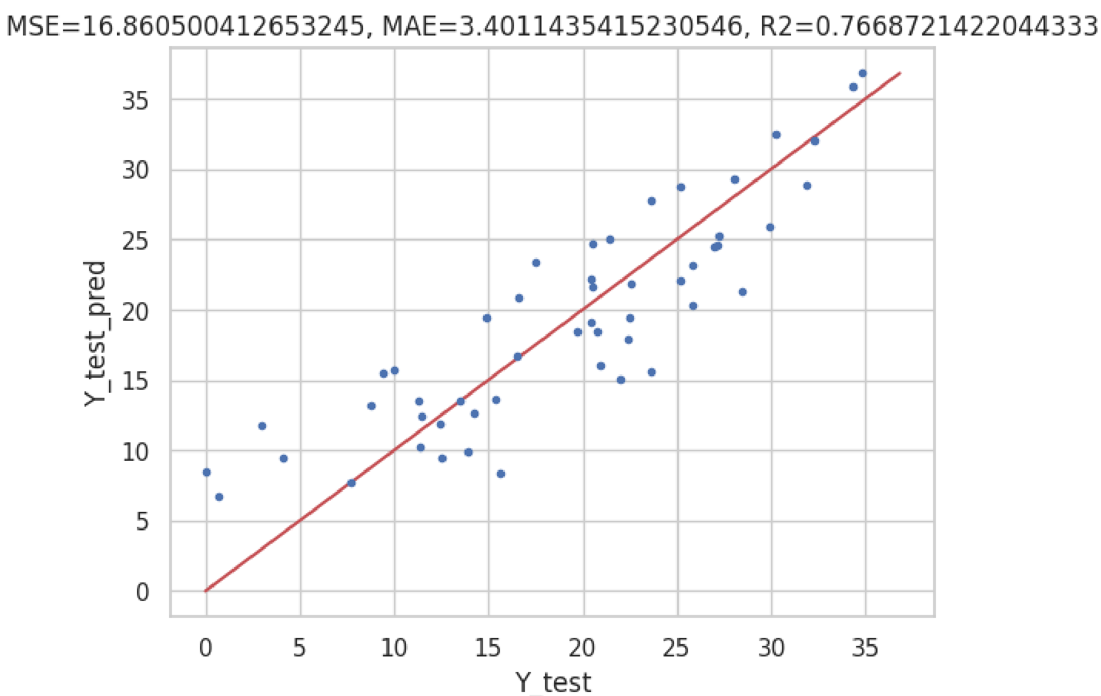


Testing results:

[ ]:
```
MSE = np.mean((Y0_test - Y0_test_pred)**2)
MAE = np.mean(np.abs(Y0_test - Y0_test_pred))
```

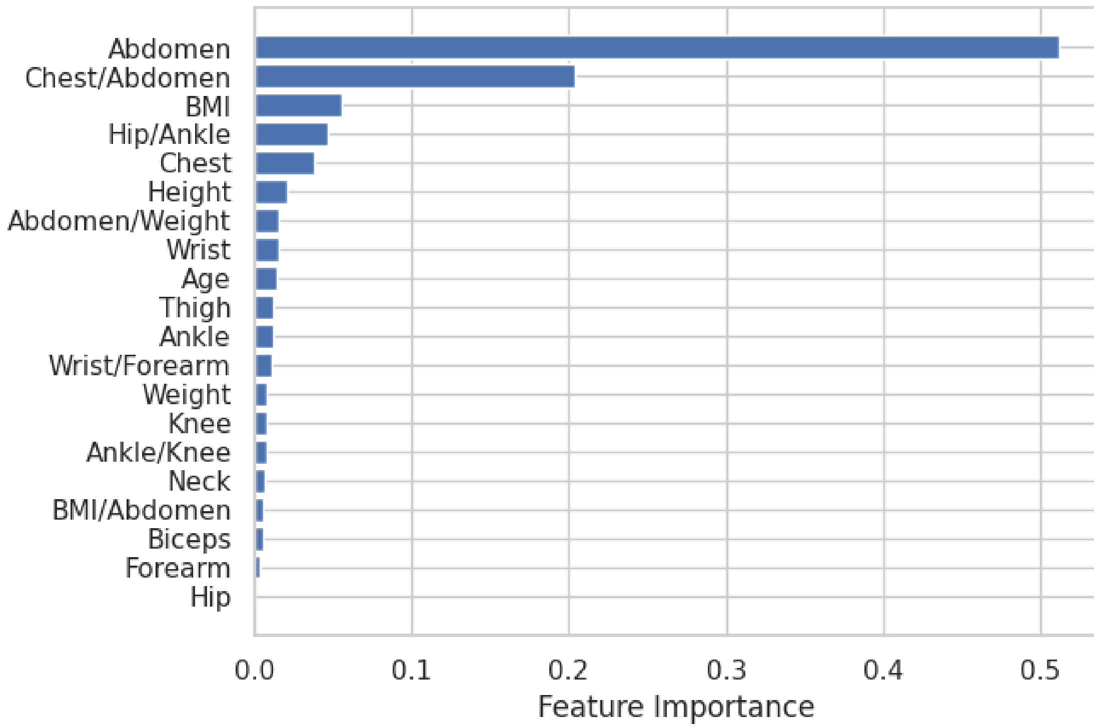
```
R2 = r2_score(Y0_test, Y0_test_pred)
#
ymax=np.max([Y0_test.max(), Y0_test_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y0_test, Y0_test_pred, '.')
plt.xlabel('Y_test')
plt.ylabel('Y_test_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[ ]: Text(0.5, 1.0, 'MSE=16.860500412653245, MAE=3.4011435415230546, R2=0.7668721422044333')



[ ]:
```
sort = model_best.feature_importances_.argsort()
plt.barh(X0.columns[sort], model_best.feature_importances_[sort])
plt.xlabel("Feature Importance")
```

[ ]: Text(0.5, 0, 'Feature Importance')



4 Part 1 Conclusion

The RF (R2=0.73) and the Linear Regression (R2=0.75) performed better between the three models we fitted by using the filtered data set. While the XGBRegressor had a similar performance than the RF.

However, when using the full data set with the new engineered features, we got the best overall performance (R2=0.76).

It is worth noticing that these results are already at the same level (or better) than the ones reported in the most upvoted project in Kaggle.

---

5 Part 2

5.1 Using auto-Sklearn

auto-sklearn frees a machine learning user from algorithm selection and hyperparameter tuning. It leverages recent advantages in Bayesian optimization, meta-learning and ensemble construction. Learn more about the technology behind auto-sklearn by reading our paper published at NeurIPS 2015.

[ ]:
```
!sudo apt-get install build-essential swig
!pip install auto-sklearn
```

[ ]:
```
!pip install -U scikit-learn
```

[ ]:
```
#!pip install --force-reinstall scipy==1.6
```

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
```

[ ]:
```
X = pd.read_csv('X.csv')
Y = pd.read_csv('Y.csv')
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
↪random_state=0)
```

[ ]:
```
from sklearn.preprocessing import StandardScaler

X_train=X_train.values

Y_train=Y_train.values

X_test=X_test.values

Y_test=Y_test.values

scaler=StandardScaler()

scaler.fit(X_train)

#normalize the features in the training set
X_train_s = scaler.transform(X_train)
#normalize the features in the test set
X_test_s = scaler.transform(X_test)
```

[ ]:
```
import autosklearn.regression

automl = autosklearn.regression.
↪AutoSklearnRegressor(time_left_for_this_task=60*30, per_run_time_limit=35,␣
↪tmp_folder="/tmp/autosklearn_regression_example_tmp")

automl.fit(X_train_s,Y_train)
```

```
[WARNING] [2023-05-04 20:34:33,546:Client-EnsembleBuilder] No runs were
available to build an ensemble from
```

[ ]:
```
AutoSklearnRegressor(ensemble_class=<class
'autosklearn.ensembles.ensemble_selection.EnsembleSelection'>,
```

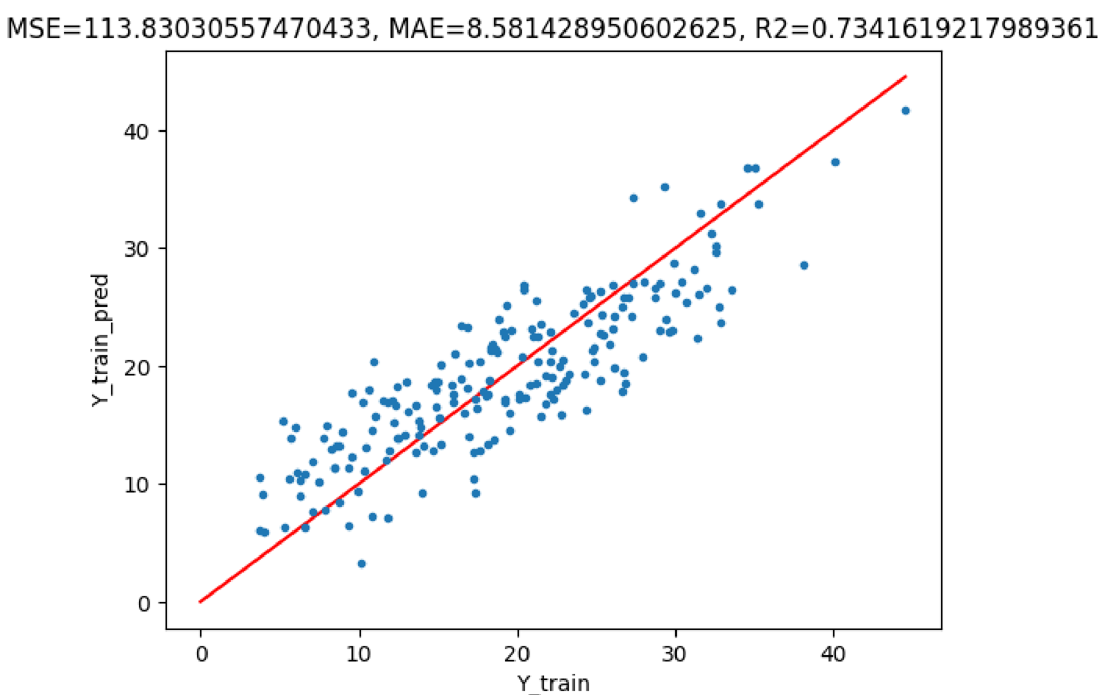
```
per_run_time_limit=35, time_left_for_this_task=1800,
tmp_folder='/tmp/autosklearn_regression_example_tmp')
```

6 Training Results:

[ ]:
```python
from sklearn.metrics import r2_score
```

[ ]:
```python
Y_train_pred = automl.predict(X_train_s)

MSE = np.mean((Y_train - Y_train_pred)**2)
MAE = np.mean(np.abs(Y_train - Y_train_pred))
R2 = r2_score(Y_train, Y_train_pred)
#
ymax=np.max([Y_train.max(), Y_train_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_train, Y_train_pred, '.')
plt.xlabel('Y_train')
plt.ylabel('Y_train_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

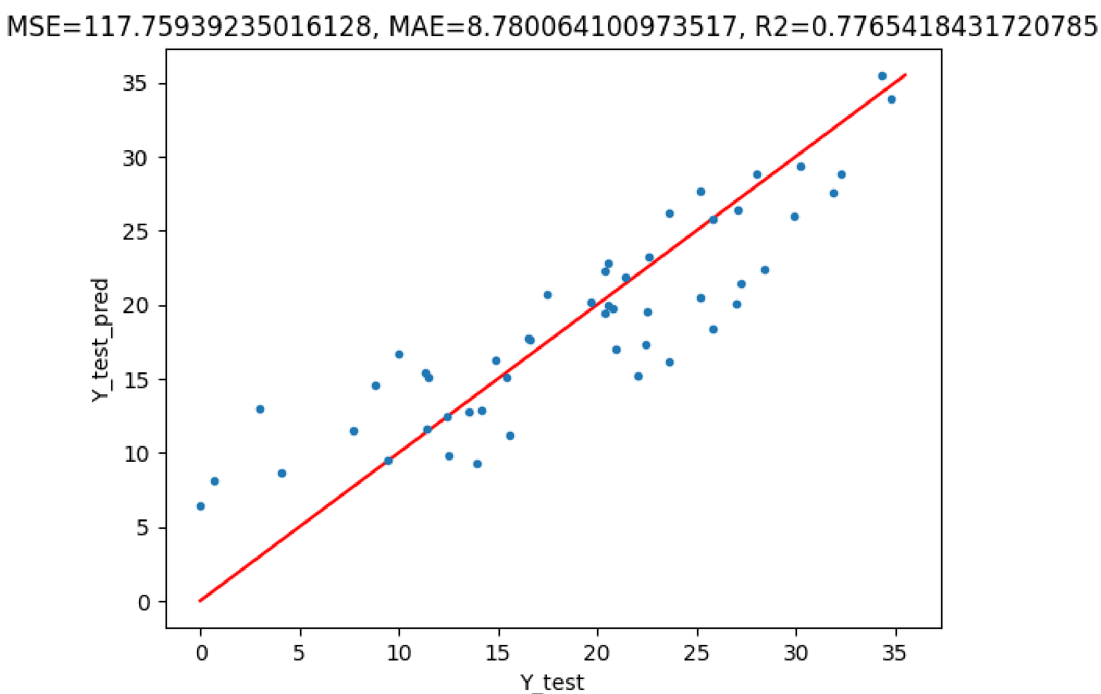
[ ]:
```
Text(0.5, 1.0, 'MSE=113.83030557470433, MAE=8.581428950602625,
R2=0.7341619217989361')
```



7 Test results:

[ ]:
```
Y_test_pred = automl.predict(X_test_s)

MSE = np.mean((Y_test - Y_test_pred)**2)
MAE = np.mean(np.abs(Y_test - Y_test_pred))
R2 = r2_score(Y_test, Y_test_pred)

ymax=np.max([Y_test.max(), Y_test_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_test, Y_test_pred, '.')
plt.xlabel('Y_test')
plt.ylabel('Y_test_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[ ]:
```
Text(0.5, 1.0, 'MSE=117.75939235016128, MAE=8.780064100973517,
R2=0.7765418431720785')
```



[ ]:
```
print(automl.leaderboard())
```

```
          rank  ensemble_weight           type      cost  duration
model_id
441          1             0.42  liblinear_svr  0.212924  0.511810
301          2             0.42  liblinear_svr  0.214227  0.693880
166          3             0.16  liblinear_svr  0.218987  0.716337
```

8 Part 2 Conclusion

The auto-sklearn produced a model that overperforms the models we fitted in part 1. However, the XGB regressor has very similar performance on the test set but also performed better on the training set.

---

9 Part 3

9.1 Neural Networks

Now we will implement neural networks to predict the bodyfat. In this notebook, we will focus on two examples: A linear regression using a network with one linear layer and no activation function; and a neural network with three layers with softmax activation.

[ ]:
```
from IPython import display
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits import mplot3d
import torch
from sklearn.model_selection import train_test_split
import pandas as pd
```

10 Data Preparation

[ ]:
```
X = pd.read_csv('X.csv')
Y = pd.read_csv('Y.csv')
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=0)
X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size=0.1,␣
 ↪random_state=0)
```

[ ]:
```
from sklearn.preprocessing import MinMaxScaler

X_train=X_train.values

Y_train=Y_train.values

X_test=X_test.values

Y_test=Y_test.values

X_val=X_val.values

Y_val=Y_val.values
```

```
scaler=MinMaxScaler()

scaler.fit(X_train)

#normalize the features in the training set
X_train_s = scaler.transform(X_train)
#normalize the features in the test set
X_test_s = scaler.transform(X_test)
#normalize the features in the validation set
X_val_s = scaler.transform(X_val)
```

[ ]:
```
from torch.utils.data import Dataset as torch_dataset
class MyDataset(torch_dataset):
    def __init__(self, X, Y):
        self.X=X
        self.Y=Y
    def __len__(self):
        #return the number of data points
        return self.X.shape[0]
    def __getitem__(self, idx):
        # return a data point (x,y) by idx (index)
        # we need to convert numpy array to torch tensor
        x=torch.tensor(self.X[idx], dtype=torch.float32)
        y=torch.tensor(self.Y[idx], dtype=torch.float32)
        return x, y
```

[ ]:
```
dataset_train = MyDataset(X_train_s, Y_train)
dataset_val = MyDataset(X_val_s, Y_val)
dataset_test = MyDataset(X_test_s, Y_test)
```

[ ]:
```
from torch.utils.data import DataLoader as torch_dataloader
dataloader_train = torch_dataloader(dataset_train, batch_size=64, shuffle=True,␣
 ↪num_workers=0)
dataloader_val = torch_dataloader(dataset_val, batch_size=64, shuffle=False,␣
 ↪num_workers=0)
dataloader_test = torch_dataloader(dataset_test, batch_size=64, shuffle=False,␣
 ↪num_workers=0)
```

[ ]:
```
for epoch in range(0, 1): # change 1 to 100 if we need to train the model for␣
 ↪100 epochs
    for batch_idx, (X, Y) in enumerate(dataloader_train):
        print(batch_idx, X.shape, Y.shape)
```

```
0 torch.Size([64, 20]) torch.Size([64, 1])
1 torch.Size([64, 20]) torch.Size([64, 1])
2 torch.Size([64, 20]) torch.Size([64, 1])
3 torch.Size([34, 20]) torch.Size([34, 1])
```

11 Creating the Network:

Our first network has only a linear layer - linear regression

[ ]:
```
import torch.nn as nn
class Net(nn.Module):
    def __init__(self, input_dim, output_dim):
        super().__init__()
        self.layer1 = nn.Linear(input_dim, output_dim)
    def forward(self, x):
        y=self.layer1(x)
        return y
```

[ ]:
```
model=Net(input_dim=20, output_dim=1)
```

[ ]:
```
x=torch.rand(1,20) # Testing
z=model(x)
```

[ ]:
```
z
```

[ ]:
```
tensor([[-0.1626]], grad_fn=<AddmmBackward0>)
```

[ ]:
```
import torch.optim as optim
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9,␣
 ↪weight_decay=1e-4)
```

[ ]:
```
def train(model, optimizer, dataloader, device, epoch):
    model.train()#set model to train mode
    loss_train=0
    for batch_idx, (X, Y) in enumerate(dataloader):
        X, Y = X.to(device), Y.to(device)
        optimizer.zero_grad()#clear the grad of each parameter, dL/dW=0
        Yp = model(X)#forward pass
        loss = torch.mean((Yp-Y)**2) # MSE loss or other loss
        loss.backward()#backward pass to get dL/dW
        optimizer.step()#update parameters:  W <= w - lr *dL/dW
        loss_train+=loss.item()#always use .item()
        if batch_idx % 1 == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                    epoch, batch_idx * X.size(0), len(dataloader.dataset),
                    100. * batch_idx / len(dataloader), loss.item()))
    loss_train/=len(dataloader)
    return loss_train
```

[ ]:
```
def test(model, dataloader, device):
    model.eval()#set model to evaluation mode
    loss_test=0
    mae_test=0
```

```
    sample_count=0
    with torch.no_grad(): # tell Pytorch not to build graph in the 'with'␣
 ↪section
        for batch_idx, (X, Y) in enumerate(dataloader):
            X, Y = X.to(device), Y.to(device)
            Yp = model(X)#forward pass
            loss_test+=torch.sum((Yp-Y)**2).item()
            mae_test+= torch.sum((Yp-Y).abs()).item()
            sample_count+=X.size(0)
    loss_test/=sample_count
    mae_test/=sample_count
    return loss_test, mae_test
```

[ ]:
```
loss_train_list=[]
loss_val_list=[]
```

[ ]:
```
device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model.to(device)
```

[ ]:
```
Net(
  (layer1): Linear(in_features=20, out_features=1, bias=True)
)
```

[ ]:
```
for epoch in range(0, 100):
    #-------- perform training --------------------------------
    loss_train=train(model, optimizer, dataloader_train, device, epoch)
    loss_train_list.append(loss_train)
    print('epoch', epoch, 'training loss:', loss_train)
    #-------- perform validation --------------------------------
    loss_val, mae_val = test(model, dataloader_val, device)
    loss_val_list.append(loss_val)
    print('epoch', epoch, 'validation loss:', loss_val)
```

```
Train Epoch: 0 [0/226 (0%)]     Loss: 438.427277
Train Epoch: 0 [64/226 (25%)]   Loss: 426.846985
Train Epoch: 0 [128/226 (50%)]  Loss: 172.717300
Train Epoch: 0 [102/226 (75%)]  Loss: 149.008667
epoch 0 training loss: 296.750057220459
epoch 0 validation loss: 49.07718599759615
Train Epoch: 1 [0/226 (0%)]     Loss: 59.186111
Train Epoch: 1 [64/226 (25%)]   Loss: 48.127182
Train Epoch: 1 [128/226 (50%)]  Loss: 82.844437
Train Epoch: 1 [102/226 (75%)]  Loss: 84.892570
epoch 1 training loss: 68.76257514953613
epoch 1 validation loss: 157.68055138221155
Train Epoch: 2 [0/226 (0%)]     Loss: 175.024750
Train Epoch: 2 [64/226 (25%)]   Loss: 212.201431
```

```
Train Epoch: 2 [128/226 (50%)]  Loss: 165.238022
Train Epoch: 2 [102/226 (75%)]  Loss: 158.276749
epoch 2 training loss: 177.68523788452148
epoch 2 validation loss: 82.39725435697116
Train Epoch: 3 [0/226 (0%)]     Loss: 99.126282
Train Epoch: 3 [64/226 (25%)]   Loss: 69.173515
Train Epoch: 3 [128/226 (50%)]  Loss: 43.467293
Train Epoch: 3 [102/226 (75%)]  Loss: 39.786293
epoch 3 training loss: 62.88834571838379
epoch 3 validation loss: 53.60506497896635
Train Epoch: 4 [0/226 (0%)]     Loss: 69.558022
Train Epoch: 4 [64/226 (25%)]   Loss: 87.527199
Train Epoch: 4 [128/226 (50%)]  Loss: 77.584183
Train Epoch: 4 [102/226 (75%)]  Loss: 68.916321
epoch 4 training loss: 75.89643096923828
epoch 4 validation loss: 74.60767540564903
Train Epoch: 5 [0/226 (0%)]     Loss: 65.657310
Train Epoch: 5 [64/226 (25%)]   Loss: 72.401619
Train Epoch: 5 [128/226 (50%)]  Loss: 48.449276
Train Epoch: 5 [102/226 (75%)]  Loss: 53.441425
epoch 5 training loss: 59.98740768432617
epoch 5 validation loss: 27.851236196664665
Train Epoch: 6 [0/226 (0%)]     Loss: 35.184692
Train Epoch: 6 [64/226 (25%)]   Loss: 32.093658
Train Epoch: 6 [128/226 (50%)]  Loss: 49.910072
Train Epoch: 6 [102/226 (75%)]  Loss: 58.999237
epoch 6 training loss: 44.04691505432129
epoch 6 validation loss: 43.32700758713942
Train Epoch: 7 [0/226 (0%)]     Loss: 50.172005
Train Epoch: 7 [64/226 (25%)]   Loss: 58.677753
Train Epoch: 7 [128/226 (50%)]  Loss: 50.845299
Train Epoch: 7 [102/226 (75%)]  Loss: 33.757607
epoch 7 training loss: 48.363165855407715
epoch 7 validation loss: 26.886922983022835
Train Epoch: 8 [0/226 (0%)]     Loss: 37.181908
Train Epoch: 8 [64/226 (25%)]   Loss: 31.258636
Train Epoch: 8 [128/226 (50%)]  Loss: 30.896641
Train Epoch: 8 [102/226 (75%)]  Loss: 27.556726
epoch 8 training loss: 31.723477840423584
epoch 8 validation loss: 31.0681880070613
Train Epoch: 9 [0/226 (0%)]     Loss: 39.203403
Train Epoch: 9 [64/226 (25%)]   Loss: 35.329533
Train Epoch: 9 [128/226 (50%)]  Loss: 34.959221
Train Epoch: 9 [102/226 (75%)]  Loss: 29.460546
epoch 9 training loss: 34.73817586898804
epoch 9 validation loss: 28.892296424278847
Train Epoch: 10 [0/226 (0%)]    Loss: 30.441645
Train Epoch: 10 [64/226 (25%)]  Loss: 32.852379
```

```
Train Epoch: 10 [128/226 (50%)] Loss: 31.140930
Train Epoch: 10 [102/226 (75%)] Loss: 17.208401
epoch 10 training loss: 27.91083860397339
epoch 10 validation loss: 22.55779090294471
Train Epoch: 11 [0/226 (0%)]    Loss: 22.740288
Train Epoch: 11 [64/226 (25%)]  Loss: 28.841412
Train Epoch: 11 [128/226 (50%)] Loss: 26.094965
Train Epoch: 11 [102/226 (75%)] Loss: 42.240475
epoch 11 training loss: 29.97928476333618
epoch 11 validation loss: 23.25456824669471
Train Epoch: 12 [0/226 (0%)]    Loss: 27.596304
Train Epoch: 12 [64/226 (25%)]  Loss: 29.142471
Train Epoch: 12 [128/226 (50%)] Loss: 26.537617
Train Epoch: 12 [102/226 (75%)] Loss: 28.391670
epoch 12 training loss: 27.917015552520752
epoch 12 validation loss: 21.5858647273137
Train Epoch: 13 [0/226 (0%)]    Loss: 21.737343
Train Epoch: 13 [64/226 (25%)]  Loss: 21.400007
Train Epoch: 13 [128/226 (50%)] Loss: 32.639778
Train Epoch: 13 [102/226 (75%)] Loss: 28.284805
epoch 13 training loss: 26.015483379364014
epoch 13 validation loss: 23.21070509690505
Train Epoch: 14 [0/226 (0%)]    Loss: 28.069588
Train Epoch: 14 [64/226 (25%)]  Loss: 20.957533
Train Epoch: 14 [128/226 (50%)] Loss: 24.846094
Train Epoch: 14 [102/226 (75%)] Loss: 28.185555
epoch 14 training loss: 25.514692306518555
epoch 14 validation loss: 21.44604022686298
Train Epoch: 15 [0/226 (0%)]    Loss: 26.242388
Train Epoch: 15 [64/226 (25%)]  Loss: 23.673763
Train Epoch: 15 [128/226 (50%)] Loss: 20.290545
Train Epoch: 15 [102/226 (75%)] Loss: 28.348873
epoch 15 training loss: 24.63889217376709
epoch 15 validation loss: 20.394444392277645
Train Epoch: 16 [0/226 (0%)]    Loss: 20.151903
Train Epoch: 16 [64/226 (25%)]  Loss: 24.380102
Train Epoch: 16 [128/226 (50%)] Loss: 25.910654
Train Epoch: 16 [102/226 (75%)] Loss: 23.526709
epoch 16 training loss: 23.492341995239258
epoch 16 validation loss: 20.36103938176082
Train Epoch: 17 [0/226 (0%)]    Loss: 18.673054
Train Epoch: 17 [64/226 (25%)]  Loss: 24.879599
Train Epoch: 17 [128/226 (50%)] Loss: 24.001400
Train Epoch: 17 [102/226 (75%)] Loss: 24.832657
epoch 17 training loss: 23.09667730331421
epoch 17 validation loss: 20.245286207932693
Train Epoch: 18 [0/226 (0%)]    Loss: 21.231089
Train Epoch: 18 [64/226 (25%)]  Loss: 27.809086
```

```
Train Epoch: 18 [128/226 (50%)] Loss: 18.210369
Train Epoch: 18 [102/226 (75%)] Loss: 22.883692
epoch 18 training loss: 22.53355884552002
epoch 18 validation loss: 19.95961115910457
Train Epoch: 19 [0/226 (0%)]    Loss: 21.753555
Train Epoch: 19 [64/226 (25%)]  Loss: 22.415184
Train Epoch: 19 [128/226 (50%)] Loss: 23.218050
Train Epoch: 19 [102/226 (75%)] Loss: 20.469790
epoch 19 training loss: 21.964144706726074
epoch 19 validation loss: 19.743248572716347
Train Epoch: 20 [0/226 (0%)]    Loss: 24.002642
Train Epoch: 20 [64/226 (25%)]  Loss: 21.899910
Train Epoch: 20 [128/226 (50%)] Loss: 21.705961
Train Epoch: 20 [102/226 (75%)] Loss: 17.986101
epoch 20 training loss: 21.398653507232666
epoch 20 validation loss: 19.643526517427883
Train Epoch: 21 [0/226 (0%)]    Loss: 21.437183
Train Epoch: 21 [64/226 (25%)]  Loss: 18.803835
Train Epoch: 21 [128/226 (50%)] Loss: 24.889429
Train Epoch: 21 [102/226 (75%)] Loss: 20.590010
epoch 21 training loss: 21.430114269256592
epoch 21 validation loss: 19.56728304349459
Train Epoch: 22 [0/226 (0%)]    Loss: 17.619724
Train Epoch: 22 [64/226 (25%)]  Loss: 19.118303
Train Epoch: 22 [128/226 (50%)] Loss: 25.570520
Train Epoch: 22 [102/226 (75%)] Loss: 24.538935
epoch 22 training loss: 21.711870670318604
epoch 22 validation loss: 19.55790006197416
Train Epoch: 23 [0/226 (0%)]    Loss: 18.809553
Train Epoch: 23 [64/226 (25%)]  Loss: 22.777794
Train Epoch: 23 [128/226 (50%)] Loss: 20.658926
Train Epoch: 23 [102/226 (75%)] Loss: 23.357805
epoch 23 training loss: 21.40101957321167
epoch 23 validation loss: 19.34041067270132
Train Epoch: 24 [0/226 (0%)]    Loss: 21.815401
Train Epoch: 24 [64/226 (25%)]  Loss: 19.497421
Train Epoch: 24 [128/226 (50%)] Loss: 23.532160
Train Epoch: 24 [102/226 (75%)] Loss: 16.843952
epoch 24 training loss: 20.42223358154297
epoch 24 validation loss: 19.107653104341946
Train Epoch: 25 [0/226 (0%)]    Loss: 20.727091
Train Epoch: 25 [64/226 (25%)]  Loss: 24.235767
Train Epoch: 25 [128/226 (50%)] Loss: 18.357283
Train Epoch: 25 [102/226 (75%)] Loss: 18.685844
epoch 25 training loss: 20.50149631500244
epoch 25 validation loss: 19.167832594651443
Train Epoch: 26 [0/226 (0%)]    Loss: 19.410069
Train Epoch: 26 [64/226 (25%)]  Loss: 20.348721
```

```
Train Epoch: 26 [128/226 (50%)] Loss: 18.996611
Train Epoch: 26 [102/226 (75%)] Loss: 26.064966
epoch 26 training loss: 21.20509147644043
epoch 26 validation loss: 19.110230665940506
Train Epoch: 27 [0/226 (0%)]    Loss: 21.481121
Train Epoch: 27 [64/226 (25%)]  Loss: 22.474449
Train Epoch: 27 [128/226 (50%)] Loss: 19.479048
Train Epoch: 27 [102/226 (75%)] Loss: 16.855274
epoch 27 training loss: 20.07247304916382
epoch 27 validation loss: 19.000728900615986
Train Epoch: 28 [0/226 (0%)]    Loss: 22.530029
Train Epoch: 28 [64/226 (25%)]  Loss: 20.513168
Train Epoch: 28 [128/226 (50%)] Loss: 17.877483
Train Epoch: 28 [102/226 (75%)] Loss: 20.405508
epoch 28 training loss: 20.331547260284424
epoch 28 validation loss: 19.35643357496995
Train Epoch: 29 [0/226 (0%)]    Loss: 18.974356
Train Epoch: 29 [64/226 (25%)]  Loss: 14.252258
Train Epoch: 29 [128/226 (50%)] Loss: 22.747671
Train Epoch: 29 [102/226 (75%)] Loss: 28.915037
epoch 29 training loss: 21.222330570220947
epoch 29 validation loss: 19.033946110652042
Train Epoch: 30 [0/226 (0%)]    Loss: 22.377058
Train Epoch: 30 [64/226 (25%)]  Loss: 20.890539
Train Epoch: 30 [128/226 (50%)] Loss: 17.791357
Train Epoch: 30 [102/226 (75%)] Loss: 18.922083
epoch 30 training loss: 19.995259284973145
epoch 30 validation loss: 18.788310124323917
Train Epoch: 31 [0/226 (0%)]    Loss: 18.976374
Train Epoch: 31 [64/226 (25%)]  Loss: 20.547079
Train Epoch: 31 [128/226 (50%)] Loss: 19.583281
Train Epoch: 31 [102/226 (75%)] Loss: 21.657646
epoch 31 training loss: 20.191094875335693
epoch 31 validation loss: 19.10341116098257
Train Epoch: 32 [0/226 (0%)]    Loss: 22.365646
Train Epoch: 32 [64/226 (25%)]  Loss: 20.168600
Train Epoch: 32 [128/226 (50%)] Loss: 17.615452
Train Epoch: 32 [102/226 (75%)] Loss: 18.517132
epoch 32 training loss: 19.666707515716553
epoch 32 validation loss: 18.844671396108772
Train Epoch: 33 [0/226 (0%)]    Loss: 22.752682
Train Epoch: 33 [64/226 (25%)]  Loss: 14.627155
Train Epoch: 33 [128/226 (50%)] Loss: 22.422693
Train Epoch: 33 [102/226 (75%)] Loss: 19.060026
epoch 33 training loss: 19.715639114379883
epoch 33 validation loss: 18.67645967923678
Train Epoch: 34 [0/226 (0%)]    Loss: 17.993217
Train Epoch: 34 [64/226 (25%)]  Loss: 23.229729
```

```
Train Epoch: 34 [128/226 (50%)] Loss: 19.081062
Train Epoch: 34 [102/226 (75%)] Loss: 17.083542
epoch 34 training loss: 19.346887588500977
epoch 34 validation loss: 18.739527775691105
Train Epoch: 35 [0/226 (0%)]    Loss: 18.118156
Train Epoch: 35 [64/226 (25%)]  Loss: 20.718016
Train Epoch: 35 [128/226 (50%)] Loss: 17.185104
Train Epoch: 35 [102/226 (75%)] Loss: 24.577549
epoch 35 training loss: 20.14970636367798
epoch 35 validation loss: 19.058235755333534
Train Epoch: 36 [0/226 (0%)]    Loss: 19.092621
Train Epoch: 36 [64/226 (25%)]  Loss: 21.092232
Train Epoch: 36 [128/226 (50%)] Loss: 18.688091
Train Epoch: 36 [102/226 (75%)] Loss: 18.686188
epoch 36 training loss: 19.389782905578613
epoch 36 validation loss: 18.818749060997597
Train Epoch: 37 [0/226 (0%)]    Loss: 20.607044
Train Epoch: 37 [64/226 (25%)]  Loss: 14.014512
Train Epoch: 37 [128/226 (50%)] Loss: 23.504755
Train Epoch: 37 [102/226 (75%)] Loss: 19.965122
epoch 37 training loss: 19.522858381271362
epoch 37 validation loss: 18.49059354341947
Train Epoch: 38 [0/226 (0%)]    Loss: 15.372845
Train Epoch: 38 [64/226 (25%)]  Loss: 18.609591
Train Epoch: 38 [128/226 (50%)] Loss: 20.262653
Train Epoch: 38 [102/226 (75%)] Loss: 26.524874
epoch 38 training loss: 20.192490577697754
epoch 38 validation loss: 18.56893803523137
Train Epoch: 39 [0/226 (0%)]    Loss: 18.491848
Train Epoch: 39 [64/226 (25%)]  Loss: 23.077343
Train Epoch: 39 [128/226 (50%)] Loss: 17.585253
Train Epoch: 39 [102/226 (75%)] Loss: 16.421856
epoch 39 training loss: 18.8940749168396
epoch 39 validation loss: 18.905929565429688
Train Epoch: 40 [0/226 (0%)]    Loss: 17.517494
Train Epoch: 40 [64/226 (25%)]  Loss: 17.676445
Train Epoch: 40 [128/226 (50%)] Loss: 21.836632
Train Epoch: 40 [102/226 (75%)] Loss: 20.417089
epoch 40 training loss: 19.361915111541748
epoch 40 validation loss: 18.824161236102764
Train Epoch: 41 [0/226 (0%)]    Loss: 17.199423
Train Epoch: 41 [64/226 (25%)]  Loss: 21.374716
Train Epoch: 41 [128/226 (50%)] Loss: 21.146833
Train Epoch: 41 [102/226 (75%)] Loss: 14.608535
epoch 41 training loss: 18.582376718521118
epoch 41 validation loss: 18.40431330754207
Train Epoch: 42 [0/226 (0%)]    Loss: 17.778210
Train Epoch: 42 [64/226 (25%)]  Loss: 19.556278
```

```
Train Epoch: 42 [128/226 (50%)] Loss: 21.620140
Train Epoch: 42 [102/226 (75%)] Loss: 15.980261
epoch 42 training loss: 18.73372220993042
epoch 42 validation loss: 18.532584557166466
Train Epoch: 43 [0/226 (0%)]    Loss: 19.919392
Train Epoch: 43 [64/226 (25%)]  Loss: 24.208935
Train Epoch: 43 [128/226 (50%)] Loss: 16.139034
Train Epoch: 43 [102/226 (75%)] Loss: 13.702708
epoch 43 training loss: 18.492517232894897
epoch 43 validation loss: 19.03176527756911
Train Epoch: 44 [0/226 (0%)]    Loss: 17.699045
Train Epoch: 44 [64/226 (25%)]  Loss: 20.919735
Train Epoch: 44 [128/226 (50%)] Loss: 21.911032
Train Epoch: 44 [102/226 (75%)] Loss: 12.151208
epoch 44 training loss: 18.170254945755005
epoch 44 validation loss: 18.636611938476562
Train Epoch: 45 [0/226 (0%)]    Loss: 24.125839
Train Epoch: 45 [64/226 (25%)]  Loss: 16.719866
Train Epoch: 45 [128/226 (50%)] Loss: 15.934834
Train Epoch: 45 [102/226 (75%)] Loss: 18.351562
epoch 45 training loss: 18.78302550315857
epoch 45 validation loss: 18.499298095703125
Train Epoch: 46 [0/226 (0%)]    Loss: 17.804585
Train Epoch: 46 [64/226 (25%)]  Loss: 20.261410
Train Epoch: 46 [128/226 (50%)] Loss: 20.959402
Train Epoch: 46 [102/226 (75%)] Loss: 13.884085
epoch 46 training loss: 18.227370262145996
epoch 46 validation loss: 18.357174213115986
Train Epoch: 47 [0/226 (0%)]    Loss: 17.985601
Train Epoch: 47 [64/226 (25%)]  Loss: 20.658039
Train Epoch: 47 [128/226 (50%)] Loss: 19.670469
Train Epoch: 47 [102/226 (75%)] Loss: 15.236310
epoch 47 training loss: 18.38760495185852
epoch 47 validation loss: 18.5262944148137
Train Epoch: 48 [0/226 (0%)]    Loss: 21.077000
Train Epoch: 48 [64/226 (25%)]  Loss: 18.444454
Train Epoch: 48 [128/226 (50%)] Loss: 17.859842
Train Epoch: 48 [102/226 (75%)] Loss: 16.288509
epoch 48 training loss: 18.41745138168335
epoch 48 validation loss: 18.568121103140022
Train Epoch: 49 [0/226 (0%)]    Loss: 17.084469
Train Epoch: 49 [64/226 (25%)]  Loss: 18.493776
Train Epoch: 49 [128/226 (50%)] Loss: 15.761977
Train Epoch: 49 [102/226 (75%)] Loss: 27.890841
epoch 49 training loss: 19.80776572227478
epoch 49 validation loss: 18.73125281700721
Train Epoch: 50 [0/226 (0%)]    Loss: 19.026676
Train Epoch: 50 [64/226 (25%)]  Loss: 18.178341
```

```
Train Epoch: 50 [128/226 (50%)] Loss: 19.989895
Train Epoch: 50 [102/226 (75%)] Loss: 16.121155
epoch 50 training loss: 18.32901668548584
epoch 50 validation loss: 18.44675973745493
Train Epoch: 51 [0/226 (0%)]    Loss: 18.264406
Train Epoch: 51 [64/226 (25%)]  Loss: 15.458733
Train Epoch: 51 [128/226 (50%)] Loss: 20.362074
Train Epoch: 51 [102/226 (75%)] Loss: 21.820038
epoch 51 training loss: 18.9763126373291
epoch 51 validation loss: 18.50433349609375
Train Epoch: 52 [0/226 (0%)]    Loss: 18.965528
Train Epoch: 52 [64/226 (25%)]  Loss: 16.985056
Train Epoch: 52 [128/226 (50%)] Loss: 21.549877
Train Epoch: 52 [102/226 (75%)] Loss: 15.391907
epoch 52 training loss: 18.223092079162598
epoch 52 validation loss: 18.659425001878006
Train Epoch: 53 [0/226 (0%)]    Loss: 19.445980
Train Epoch: 53 [64/226 (25%)]  Loss: 15.857219
Train Epoch: 53 [128/226 (50%)] Loss: 16.131294
Train Epoch: 53 [102/226 (75%)] Loss: 28.510376
epoch 53 training loss: 19.986217260360718
epoch 53 validation loss: 18.27084702711839
Train Epoch: 54 [0/226 (0%)]    Loss: 17.972343
Train Epoch: 54 [64/226 (25%)]  Loss: 17.795713
Train Epoch: 54 [128/226 (50%)] Loss: 19.528986
Train Epoch: 54 [102/226 (75%)] Loss: 18.613058
epoch 54 training loss: 18.477525234222412
epoch 54 validation loss: 18.62693669245793
Train Epoch: 55 [0/226 (0%)]    Loss: 22.169605
Train Epoch: 55 [64/226 (25%)]  Loss: 14.946848
Train Epoch: 55 [128/226 (50%)] Loss: 18.620028
Train Epoch: 55 [102/226 (75%)] Loss: 17.821430
epoch 55 training loss: 18.389477729797363
epoch 55 validation loss: 18.862872783954327
Train Epoch: 56 [0/226 (0%)]    Loss: 19.863659
Train Epoch: 56 [64/226 (25%)]  Loss: 20.035419
Train Epoch: 56 [128/226 (50%)] Loss: 15.023476
Train Epoch: 56 [102/226 (75%)] Loss: 19.660967
epoch 56 training loss: 18.645880222320557
epoch 56 validation loss: 18.47642282339243
Train Epoch: 57 [0/226 (0%)]    Loss: 18.304974
Train Epoch: 57 [64/226 (25%)]  Loss: 15.300855
Train Epoch: 57 [128/226 (50%)] Loss: 20.340982
Train Epoch: 57 [102/226 (75%)] Loss: 20.492899
epoch 57 training loss: 18.60992741584778
epoch 57 validation loss: 18.487832876352165
Train Epoch: 58 [0/226 (0%)]    Loss: 19.124105
Train Epoch: 58 [64/226 (25%)]  Loss: 19.309586
```

```
Train Epoch: 58 [128/226 (50%)] Loss: 18.069326
Train Epoch: 58 [102/226 (75%)] Loss: 15.422044
epoch 58 training loss: 17.98126530647278
epoch 58 validation loss: 18.485411423903244
Train Epoch: 59 [0/226 (0%)]    Loss: 18.395281
Train Epoch: 59 [64/226 (25%)]  Loss: 16.345280
Train Epoch: 59 [128/226 (50%)] Loss: 18.861061
Train Epoch: 59 [102/226 (75%)] Loss: 21.015028
epoch 59 training loss: 18.654162406921387
epoch 59 validation loss: 18.43510202261118
Train Epoch: 60 [0/226 (0%)]    Loss: 12.385084
Train Epoch: 60 [64/226 (25%)]  Loss: 18.421427
Train Epoch: 60 [128/226 (50%)] Loss: 20.759195
Train Epoch: 60 [102/226 (75%)] Loss: 24.381744
epoch 60 training loss: 18.986862659454346
epoch 60 validation loss: 18.35388418344351
Train Epoch: 61 [0/226 (0%)]    Loss: 17.616331
Train Epoch: 61 [64/226 (25%)]  Loss: 17.100521
Train Epoch: 61 [128/226 (50%)] Loss: 17.468212
Train Epoch: 61 [102/226 (75%)] Loss: 23.005186
epoch 61 training loss: 18.79756259918213
epoch 61 validation loss: 18.433094904972958
Train Epoch: 62 [0/226 (0%)]    Loss: 21.008919
Train Epoch: 62 [64/226 (25%)]  Loss: 15.490452
Train Epoch: 62 [128/226 (50%)] Loss: 15.612522
Train Epoch: 62 [102/226 (75%)] Loss: 22.918955
epoch 62 training loss: 18.75771188735962
epoch 62 validation loss: 18.403218195988583
Train Epoch: 63 [0/226 (0%)]    Loss: 18.486523
Train Epoch: 63 [64/226 (25%)]  Loss: 19.507244
Train Epoch: 63 [128/226 (50%)] Loss: 17.291702
Train Epoch: 63 [102/226 (75%)] Loss: 16.924112
epoch 63 training loss: 18.052395343780518
epoch 63 validation loss: 18.517674372746395
Train Epoch: 64 [0/226 (0%)]    Loss: 18.430828
Train Epoch: 64 [64/226 (25%)]  Loss: 19.008528
Train Epoch: 64 [128/226 (50%)] Loss: 20.167568
Train Epoch: 64 [102/226 (75%)] Loss: 12.473090
epoch 64 training loss: 17.5200035572052
epoch 64 validation loss: 18.23463087815505
Train Epoch: 65 [0/226 (0%)]    Loss: 15.151381
Train Epoch: 65 [64/226 (25%)]  Loss: 21.937620
Train Epoch: 65 [128/226 (50%)] Loss: 16.267424
Train Epoch: 65 [102/226 (75%)] Loss: 20.024063
epoch 65 training loss: 18.34512186050415
epoch 65 validation loss: 18.328768216646633
Train Epoch: 66 [0/226 (0%)]    Loss: 16.651691
Train Epoch: 66 [64/226 (25%)]  Loss: 20.425133
```

```
Train Epoch: 66 [128/226 (50%)] Loss: 16.598694
Train Epoch: 66 [102/226 (75%)] Loss: 19.410402
epoch 66 training loss: 18.271480083465576
epoch 66 validation loss: 18.546383197490986
Train Epoch: 67 [0/226 (0%)]    Loss: 18.376322
Train Epoch: 67 [64/226 (25%)]  Loss: 17.710598
Train Epoch: 67 [128/226 (50%)] Loss: 19.215809
Train Epoch: 67 [102/226 (75%)] Loss: 16.076431
epoch 67 training loss: 17.84478998184204
epoch 67 validation loss: 18.35998769906851
Train Epoch: 68 [0/226 (0%)]    Loss: 18.620081
Train Epoch: 68 [64/226 (25%)]  Loss: 18.037062
Train Epoch: 68 [128/226 (50%)] Loss: 15.131589
Train Epoch: 68 [102/226 (75%)] Loss: 22.648209
epoch 68 training loss: 18.609235048294067
epoch 68 validation loss: 18.15412315955529
Train Epoch: 69 [0/226 (0%)]    Loss: 23.062403
Train Epoch: 69 [64/226 (25%)]  Loss: 15.814285
Train Epoch: 69 [128/226 (50%)] Loss: 14.145689
Train Epoch: 69 [102/226 (75%)] Loss: 20.193027
epoch 69 training loss: 18.30385112762451
epoch 69 validation loss: 18.337042001577522
Train Epoch: 70 [0/226 (0%)]    Loss: 19.574188
Train Epoch: 70 [64/226 (25%)]  Loss: 14.899645
Train Epoch: 70 [128/226 (50%)] Loss: 18.464748
Train Epoch: 70 [102/226 (75%)] Loss: 20.035732
epoch 70 training loss: 18.24357843399048
epoch 70 validation loss: 18.40901418832632
Train Epoch: 71 [0/226 (0%)]    Loss: 22.442959
Train Epoch: 71 [64/226 (25%)]  Loss: 16.040888
Train Epoch: 71 [128/226 (50%)] Loss: 16.551718
Train Epoch: 71 [102/226 (75%)] Loss: 15.950897
epoch 71 training loss: 17.746615409851074
epoch 71 validation loss: 18.3377192570613
Train Epoch: 72 [0/226 (0%)]    Loss: 14.917793
Train Epoch: 72 [64/226 (25%)]  Loss: 18.839041
Train Epoch: 72 [128/226 (50%)] Loss: 19.570501
Train Epoch: 72 [102/226 (75%)] Loss: 19.242399
epoch 72 training loss: 18.142433643341064
epoch 72 validation loss: 18.28749788724459
Train Epoch: 73 [0/226 (0%)]    Loss: 16.763350
Train Epoch: 73 [64/226 (25%)]  Loss: 17.417706
Train Epoch: 73 [128/226 (50%)] Loss: 19.821018
Train Epoch: 73 [102/226 (75%)] Loss: 17.962261
epoch 73 training loss: 17.99108362197876
epoch 73 validation loss: 18.1726801945613
Train Epoch: 74 [0/226 (0%)]    Loss: 17.441433
Train Epoch: 74 [64/226 (25%)]  Loss: 17.018833
```

```
Train Epoch: 74 [128/226 (50%)] Loss: 18.583403
Train Epoch: 74 [102/226 (75%)] Loss: 19.424063
epoch 74 training loss: 18.11693286895752
epoch 74 validation loss: 18.51279977651743
Train Epoch: 75 [0/226 (0%)]    Loss: 15.349744
Train Epoch: 75 [64/226 (25%)]  Loss: 20.364765
Train Epoch: 75 [128/226 (50%)] Loss: 18.320070
Train Epoch: 75 [102/226 (75%)] Loss: 17.504858
epoch 75 training loss: 17.884859323501587
epoch 75 validation loss: 18.45157681978666
Train Epoch: 76 [0/226 (0%)]    Loss: 15.686245
Train Epoch: 76 [64/226 (25%)]  Loss: 18.171902
Train Epoch: 76 [128/226 (50%)] Loss: 18.805475
Train Epoch: 76 [102/226 (75%)] Loss: 19.970644
epoch 76 training loss: 18.15856647491455
epoch 76 validation loss: 18.50492154634916
Train Epoch: 77 [0/226 (0%)]    Loss: 17.414474
Train Epoch: 77 [64/226 (25%)]  Loss: 17.563763
Train Epoch: 77 [128/226 (50%)] Loss: 15.783282
Train Epoch: 77 [102/226 (75%)] Loss: 23.164835
epoch 77 training loss: 18.48158860206604
epoch 77 validation loss: 18.24020033616286
Train Epoch: 78 [0/226 (0%)]    Loss: 14.326753
Train Epoch: 78 [64/226 (25%)]  Loss: 17.372753
Train Epoch: 78 [128/226 (50%)] Loss: 23.636919
Train Epoch: 78 [102/226 (75%)] Loss: 14.550682
epoch 78 training loss: 17.471776723861694
epoch 78 validation loss: 18.096878051757812
Train Epoch: 79 [0/226 (0%)]    Loss: 17.822861
Train Epoch: 79 [64/226 (25%)]  Loss: 20.421444
Train Epoch: 79 [128/226 (50%)] Loss: 15.098343
Train Epoch: 79 [102/226 (75%)] Loss: 18.580263
epoch 79 training loss: 17.980727672576904
epoch 79 validation loss: 18.154876708984375
Train Epoch: 80 [0/226 (0%)]    Loss: 15.963499
Train Epoch: 80 [64/226 (25%)]  Loss: 21.413767
Train Epoch: 80 [128/226 (50%)] Loss: 16.515471
Train Epoch: 80 [102/226 (75%)] Loss: 17.648075
epoch 80 training loss: 17.885202884674072
epoch 80 validation loss: 18.271555973933292
Train Epoch: 81 [0/226 (0%)]    Loss: 15.645393
Train Epoch: 81 [64/226 (25%)]  Loss: 17.511497
Train Epoch: 81 [128/226 (50%)] Loss: 18.872583
Train Epoch: 81 [102/226 (75%)] Loss: 20.569479
epoch 81 training loss: 18.149738311767578
epoch 81 validation loss: 18.167394784780647
Train Epoch: 82 [0/226 (0%)]    Loss: 19.510609
Train Epoch: 82 [64/226 (25%)]  Loss: 19.248913
```

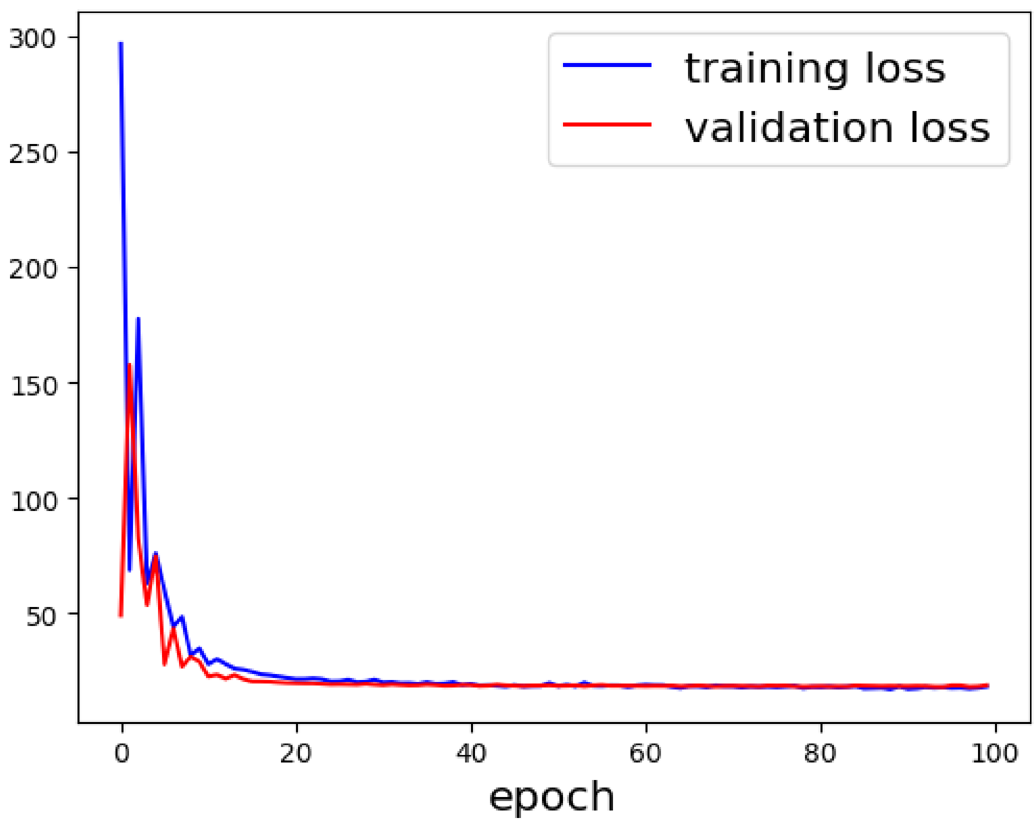
```
Train Epoch: 82 [128/226 (50%)] Loss: 15.135402
Train Epoch: 82 [102/226 (75%)] Loss: 17.125343
epoch 82 training loss: 17.755066633224487
epoch 82 validation loss: 18.211412869966946
Train Epoch: 83 [0/226 (0%)]    Loss: 17.268372
Train Epoch: 83 [64/226 (25%)]  Loss: 19.934530
Train Epoch: 83 [128/226 (50%)] Loss: 15.534992
Train Epoch: 83 [102/226 (75%)] Loss: 19.302868
epoch 83 training loss: 18.01019048690796
epoch 83 validation loss: 18.1411625788762
Train Epoch: 84 [0/226 (0%)]    Loss: 18.067135
Train Epoch: 84 [64/226 (25%)]  Loss: 15.990441
Train Epoch: 84 [128/226 (50%)] Loss: 17.032272
Train Epoch: 84 [102/226 (75%)] Loss: 22.100870
epoch 84 training loss: 18.297679662704468
epoch 84 validation loss: 18.553519615760216
Train Epoch: 85 [0/226 (0%)]    Loss: 21.084366
Train Epoch: 85 [64/226 (25%)]  Loss: 14.356730
Train Epoch: 85 [128/226 (50%)] Loss: 19.931976
Train Epoch: 85 [102/226 (75%)] Loss: 14.034148
epoch 85 training loss: 17.351804971694946
epoch 85 validation loss: 18.451089712289665
Train Epoch: 86 [0/226 (0%)]    Loss: 17.704123
Train Epoch: 86 [64/226 (25%)]  Loss: 17.387833
Train Epoch: 86 [128/226 (50%)] Loss: 19.461687
Train Epoch: 86 [102/226 (75%)] Loss: 15.347856
epoch 86 training loss: 17.475374460220337
epoch 86 validation loss: 18.37881587101863
Train Epoch: 87 [0/226 (0%)]    Loss: 17.521486
Train Epoch: 87 [64/226 (25%)]  Loss: 17.522436
Train Epoch: 87 [128/226 (50%)] Loss: 18.704369
Train Epoch: 87 [102/226 (75%)] Loss: 16.653721
epoch 87 training loss: 17.600502967834473
epoch 87 validation loss: 18.28305171086238
Train Epoch: 88 [0/226 (0%)]    Loss: 18.363014
Train Epoch: 88 [64/226 (25%)]  Loss: 20.842466
Train Epoch: 88 [128/226 (50%)] Loss: 17.569733
Train Epoch: 88 [102/226 (75%)] Loss: 11.115403
epoch 88 training loss: 16.972654104232788
epoch 88 validation loss: 18.3645512507512
Train Epoch: 89 [0/226 (0%)]    Loss: 15.401764
Train Epoch: 89 [64/226 (25%)]  Loss: 17.789560
Train Epoch: 89 [128/226 (50%)] Loss: 17.574385
Train Epoch: 89 [102/226 (75%)] Loss: 22.129005
epoch 89 training loss: 18.223678588867188
epoch 89 validation loss: 18.29684565617488
Train Epoch: 90 [0/226 (0%)]    Loss: 22.182081
Train Epoch: 90 [64/226 (25%)]  Loss: 19.342726
```

```
Train Epoch: 90 [128/226 (50%)] Loss: 14.124623
Train Epoch: 90 [102/226 (75%)] Loss: 12.796719
epoch 90 training loss: 17.111537218093872
epoch 90 validation loss: 18.27407015286959
Train Epoch: 91 [0/226 (0%)]    Loss: 18.307035
Train Epoch: 91 [64/226 (25%)]  Loss: 14.918131
Train Epoch: 91 [128/226 (50%)] Loss: 21.762896
Train Epoch: 91 [102/226 (75%)] Loss: 14.476099
epoch 91 training loss: 17.366040229797363
epoch 91 validation loss: 18.380564762995792
Train Epoch: 92 [0/226 (0%)]    Loss: 17.657825
Train Epoch: 92 [64/226 (25%)]  Loss: 17.525553
Train Epoch: 92 [128/226 (50%)] Loss: 16.831245
Train Epoch: 92 [102/226 (75%)] Loss: 19.678329
epoch 92 training loss: 17.923238277435303
epoch 92 validation loss: 18.30438232421875
Train Epoch: 93 [0/226 (0%)]    Loss: 19.406467
Train Epoch: 93 [64/226 (25%)]  Loss: 17.455385
Train Epoch: 93 [128/226 (50%)] Loss: 16.061726
Train Epoch: 93 [102/226 (75%)] Loss: 17.529234
epoch 93 training loss: 17.613203048706055
epoch 93 validation loss: 18.05611360990084
Train Epoch: 94 [0/226 (0%)]    Loss: 16.895367
Train Epoch: 94 [64/226 (25%)]  Loss: 20.768776
Train Epoch: 94 [128/226 (50%)] Loss: 15.521358
Train Epoch: 94 [102/226 (75%)] Loss: 18.173195
epoch 94 training loss: 17.83967399597168
epoch 94 validation loss: 17.976587148813103
Train Epoch: 95 [0/226 (0%)]    Loss: 18.217306
Train Epoch: 95 [64/226 (25%)]  Loss: 19.539854
Train Epoch: 95 [128/226 (50%)] Loss: 16.640230
Train Epoch: 95 [102/226 (75%)] Loss: 15.883245
epoch 95 training loss: 17.57015895843506
epoch 95 validation loss: 18.628360454852764
Train Epoch: 96 [0/226 (0%)]    Loss: 16.190273
Train Epoch: 96 [64/226 (25%)]  Loss: 13.643373
Train Epoch: 96 [128/226 (50%)] Loss: 23.790659
Train Epoch: 96 [102/226 (75%)] Loss: 17.123550
epoch 96 training loss: 17.68696403503418
epoch 96 validation loss: 18.62372295673077
Train Epoch: 97 [0/226 (0%)]    Loss: 18.887373
Train Epoch: 97 [64/226 (25%)]  Loss: 18.450541
Train Epoch: 97 [128/226 (50%)] Loss: 17.973175
Train Epoch: 97 [102/226 (75%)] Loss: 13.752284
epoch 97 training loss: 17.265843152999878
epoch 97 validation loss: 18.029528104341946
Train Epoch: 98 [0/226 (0%)]    Loss: 13.956959
Train Epoch: 98 [64/226 (25%)]  Loss: 16.103519
```

```
Train Epoch: 98 [128/226 (50%)] Loss: 23.247961
Train Epoch: 98 [102/226 (75%)] Loss: 17.088894
epoch 98 training loss: 17.5993332862854
epoch 98 validation loss: 18.169056819035458
Train Epoch: 99 [0/226 (0%)]    Loss: 15.164419
Train Epoch: 99 [64/226 (25%)]  Loss: 15.725915
Train Epoch: 99 [128/226 (50%)] Loss: 20.027159
Train Epoch: 99 [102/226 (75%)] Loss: 21.576023
epoch 99 training loss: 18.12337899208069
epoch 99 validation loss: 18.69940655048077
```

[ ]:
```
fig, ax = plt.subplots()
ax.plot(np.arange(0,len(loss_train_list)), loss_train_list, '-b',␣
 ↪label='training loss')
ax.plot(np.arange(0,len(loss_val_list)), loss_val_list, '-r', label='validation␣
 ↪loss')
ax.set_xlabel('epoch',fontsize=16)
ax.legend(fontsize=16)
```

[ ]: <matplotlib.legend.Legend at 0x7f7c7c49a9b0>

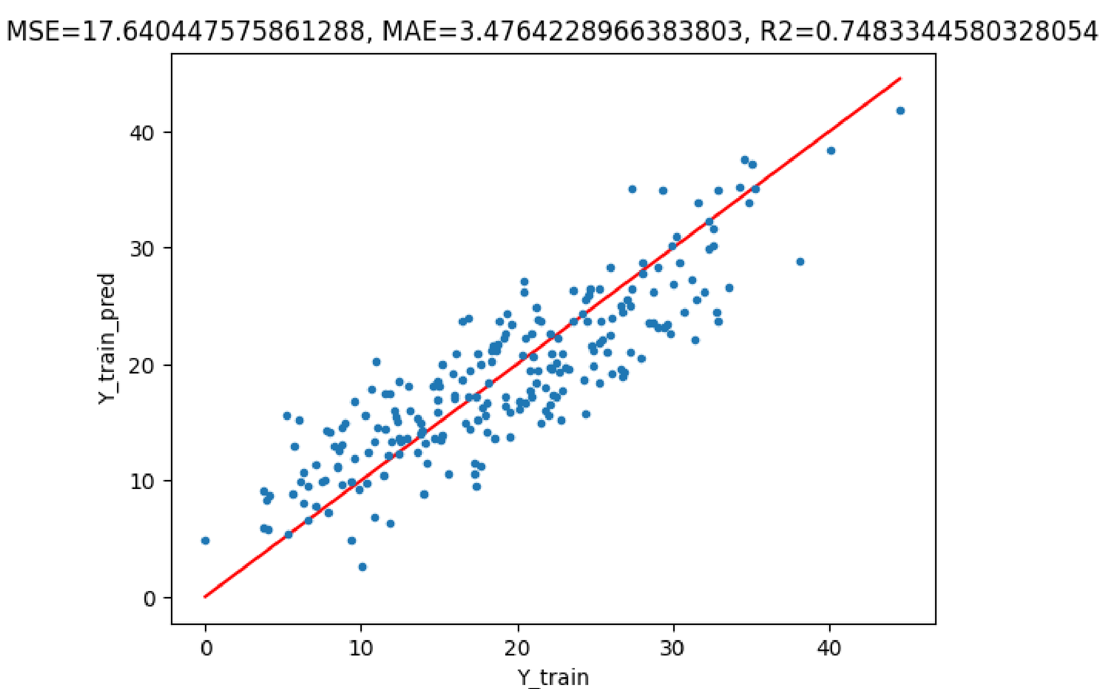


12 Training results:

```
[ ]: X_train_t=torch.tensor(X_train_s, dtype=torch.float32) #convert numpy array to␣
     ↪torch tensor
     #the input to model should be torch tensor
     Y_train_pred_t=model(X_train_t.to(device))
     Y_train_pred=Y_train_pred_t.detach().cpu().numpy() #convert torch tensor to␣
     ↪numpy array
```

```
[ ]: from sklearn.metrics import r2_score

     MSE = np.mean((Y_train - Y_train_pred)**2)
     MAE = np.mean(np.abs(Y_train - Y_train_pred))
     R2 = r2_score(Y_train, Y_train_pred)
     #
     ymax=np.max([Y_train.max(), Y_train_pred.max()])
     plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
     plt.plot(Y_train, Y_train_pred, '.')
     plt.xlabel('Y_train')
     plt.ylabel('Y_train_pred')
     plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

```
[ ]: Text(0.5, 1.0, 'MSE=17.640447575861288, MAE=3.4764228966383803,
     R2=0.7483344580328054')
```

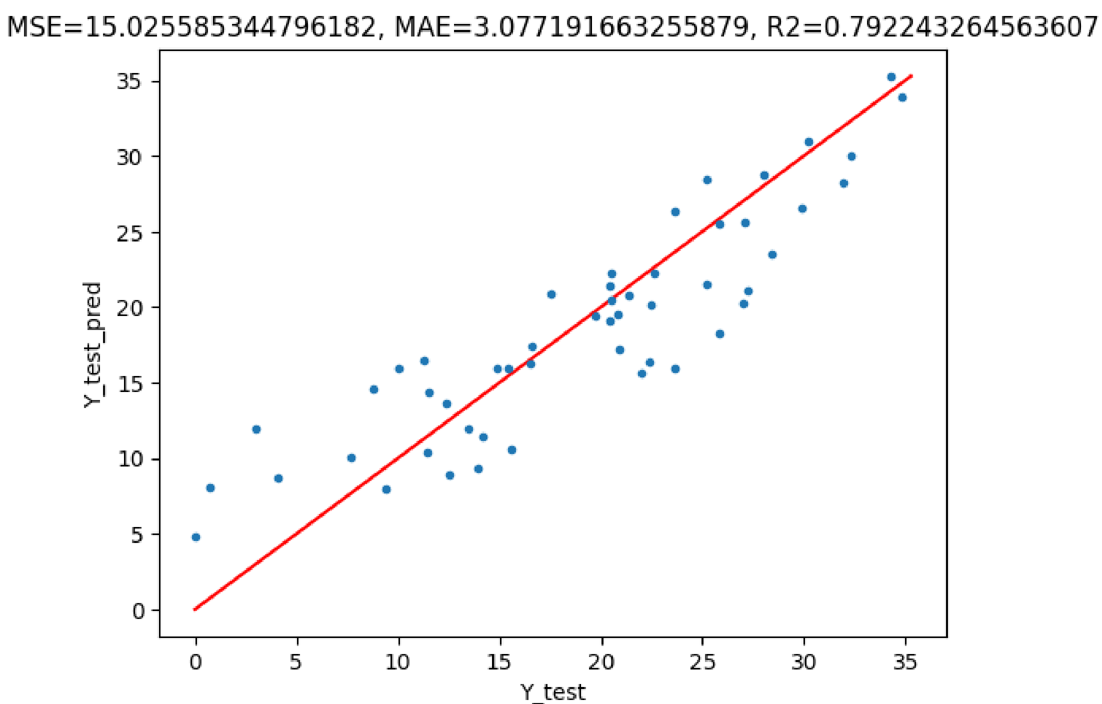


13 Testing results:

```
[ ]: X_test_t=torch.tensor(X_test_s, dtype=torch.float32) #convert numpy array to␣
     ↪torch tensor
     #the input to model should be torch tensor
     Y_test_pred_t=model(X_test_t.to(device))
     Y_test_pred=Y_test_pred_t.detach().cpu().numpy() #convert torch tensor to numpy␣
     ↪array
```

```
[ ]: MSE = np.mean((Y_test - Y_test_pred)**2)
     MAE = np.mean(np.abs(Y_test - Y_test_pred))
     R2 = r2_score(Y_test, Y_test_pred)
     #
     ymax=np.max([Y_test.max(), Y_test_pred.max()])
     plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
     plt.plot(Y_test, Y_test_pred, '.')
     plt.xlabel('Y_test')
     plt.ylabel('Y_test_pred')
     plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

```
[ ]: Text(0.5, 1.0, 'MSE=15.025585344796182, MAE=3.077191663255879,
     R2=0.792243264563607')
```



14 Creating a second Neural Network:

This one has two layers with ReLu activation

[ ]:
```
import torch.nn.functional as nnF
import torch.nn as nn

class Net_nonlin(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer1 = nn.Linear(20, 32)
        self.layer2 = nn.Linear(32, 1)
    def forward(self, x):
        x=self.layer1(x)
        x=nnF.relu(x)
        y=self.layer2(x)
        return y
```

[ ]:
```
model=Net_nonlin()
device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model.to(device)
```

[ ]:
```
Net_nonlin(
  (layer1): Linear(in_features=20, out_features=32, bias=True)
  (layer2): Linear(in_features=32, out_features=1, bias=True)
)
```

[ ]:
```
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9,␣
 ↪weight_decay=1e-4)

loss_train_list=[]
loss_val_list=[]

for epoch in range(0, 100):
    #-------- perform training --------------------------------
    loss_train=train(model, optimizer, dataloader_train, device, epoch)
    loss_train_list.append(loss_train)
    print('epoch', epoch, 'training loss:', loss_train)
    #-------- perform validation --------------------------------
    loss_val, mae_val = test(model, dataloader_val, device)
    loss_val_list.append(loss_val)
    print('epoch', epoch, 'validation loss:', loss_val)
```

```
Train Epoch: 0 [0/226 (0%)]     Loss: 403.624939
Train Epoch: 0 [64/226 (25%)]   Loss: 405.216156
Train Epoch: 0 [128/226 (50%)]  Loss: 222.479004
Train Epoch: 0 [102/226 (75%)]  Loss: 215.160233
epoch 0 training loss: 311.6200828552246
```

```
epoch 0 validation loss: 56.3138192983774
Train Epoch: 1 [0/226 (0%)]     Loss: 78.604492
Train Epoch: 1 [64/226 (25%)]   Loss: 133.366943
Train Epoch: 1 [128/226 (50%)]  Loss: 146.168518
Train Epoch: 1 [102/226 (75%)]  Loss: 59.227989
epoch 1 training loss: 104.34198570251465
epoch 1 validation loss: 139.46858097956732
Train Epoch: 2 [0/226 (0%)]     Loss: 141.246704
Train Epoch: 2 [64/226 (25%)]   Loss: 44.713570
Train Epoch: 2 [128/226 (50%)]  Loss: 89.262444
Train Epoch: 2 [102/226 (75%)]  Loss: 111.111816
epoch 2 training loss: 96.58363342285156
epoch 2 validation loss: 29.89543738731971
Train Epoch: 3 [0/226 (0%)]     Loss: 32.224190
Train Epoch: 3 [64/226 (25%)]   Loss: 46.640221
Train Epoch: 3 [128/226 (50%)]  Loss: 45.279438
Train Epoch: 3 [102/226 (75%)]  Loss: 34.031525
epoch 3 training loss: 39.543843269348145
epoch 3 validation loss: 54.59147291917067
Train Epoch: 4 [0/226 (0%)]     Loss: 48.467072
Train Epoch: 4 [64/226 (25%)]   Loss: 30.458572
Train Epoch: 4 [128/226 (50%)]  Loss: 24.207624
Train Epoch: 4 [102/226 (75%)]  Loss: 61.797249
epoch 4 training loss: 41.23262929916382
epoch 4 validation loss: 21.85590069110577
Train Epoch: 5 [0/226 (0%)]     Loss: 21.350796
Train Epoch: 5 [64/226 (25%)]   Loss: 44.184872
Train Epoch: 5 [128/226 (50%)]  Loss: 30.467928
Train Epoch: 5 [102/226 (75%)]  Loss: 21.049837
epoch 5 training loss: 29.263358116149902
epoch 5 validation loss: 26.568805401141827
Train Epoch: 6 [0/226 (0%)]     Loss: 31.670181
Train Epoch: 6 [64/226 (25%)]   Loss: 24.741564
Train Epoch: 6 [128/226 (50%)]  Loss: 26.914530
Train Epoch: 6 [102/226 (75%)]  Loss: 26.047626
epoch 6 training loss: 27.343475341796875
epoch 6 validation loss: 32.34635573167067
Train Epoch: 7 [0/226 (0%)]     Loss: 25.721807
Train Epoch: 7 [64/226 (25%)]   Loss: 20.424927
Train Epoch: 7 [128/226 (50%)]  Loss: 33.496922
Train Epoch: 7 [102/226 (75%)]  Loss: 25.048935
epoch 7 training loss: 26.173147678375244
epoch 7 validation loss: 21.45086669921875
Train Epoch: 8 [0/226 (0%)]     Loss: 23.339867
Train Epoch: 8 [64/226 (25%)]   Loss: 27.356657
Train Epoch: 8 [128/226 (50%)]  Loss: 20.553162
Train Epoch: 8 [102/226 (75%)]  Loss: 20.339718
epoch 8 training loss: 22.897350788116455
```

```
epoch 8 validation loss: 21.112337552584133
Train Epoch: 9 [0/226 (0%)]     Loss: 28.503969
Train Epoch: 9 [64/226 (25%)]   Loss: 18.471142
Train Epoch: 9 [128/226 (50%)]  Loss: 17.449516
Train Epoch: 9 [102/226 (75%)]  Loss: 26.575283
epoch 9 training loss: 22.749977588653564
epoch 9 validation loss: 23.34210205078125
Train Epoch: 10 [0/226 (0%)]    Loss: 23.967669
Train Epoch: 10 [64/226 (25%)]  Loss: 18.792715
Train Epoch: 10 [128/226 (50%)] Loss: 19.235193
Train Epoch: 10 [102/226 (75%)] Loss: 22.087542
epoch 10 training loss: 21.020779609680176
epoch 10 validation loss: 19.450755192683292
Train Epoch: 11 [0/226 (0%)]    Loss: 17.781845
Train Epoch: 11 [64/226 (25%)]  Loss: 17.001421
Train Epoch: 11 [128/226 (50%)] Loss: 20.998085
Train Epoch: 11 [102/226 (75%)] Loss: 25.131281
epoch 11 training loss: 20.228157997131348
epoch 11 validation loss: 20.495394193209133
Train Epoch: 12 [0/226 (0%)]    Loss: 22.963959
Train Epoch: 12 [64/226 (25%)]  Loss: 18.294260
Train Epoch: 12 [128/226 (50%)] Loss: 19.872112
Train Epoch: 12 [102/226 (75%)] Loss: 18.324818
epoch 12 training loss: 19.863787174224854
epoch 12 validation loss: 20.988807091346153
Train Epoch: 13 [0/226 (0%)]    Loss: 17.697758
Train Epoch: 13 [64/226 (25%)]  Loss: 16.842274
Train Epoch: 13 [128/226 (50%)] Loss: 22.824081
Train Epoch: 13 [102/226 (75%)] Loss: 21.075541
epoch 13 training loss: 19.60991334915161
epoch 13 validation loss: 19.385743361253006
Train Epoch: 14 [0/226 (0%)]    Loss: 17.412157
Train Epoch: 14 [64/226 (25%)]  Loss: 21.292030
Train Epoch: 14 [128/226 (50%)] Loss: 24.106323
Train Epoch: 14 [102/226 (75%)] Loss: 15.763782
epoch 14 training loss: 19.64357304573059
epoch 14 validation loss: 19.292942927433895
Train Epoch: 15 [0/226 (0%)]    Loss: 20.903925
Train Epoch: 15 [64/226 (25%)]  Loss: 19.434202
Train Epoch: 15 [128/226 (50%)] Loss: 20.278944
Train Epoch: 15 [102/226 (75%)] Loss: 13.535261
epoch 15 training loss: 18.53808307647705
epoch 15 validation loss: 25.110182542067307
Train Epoch: 16 [0/226 (0%)]    Loss: 22.196356
Train Epoch: 16 [64/226 (25%)]  Loss: 24.391600
Train Epoch: 16 [128/226 (50%)] Loss: 19.020372
Train Epoch: 16 [102/226 (75%)] Loss: 11.613925
epoch 16 training loss: 19.30556321144104
```

```
epoch 16 validation loss: 18.937953068659855
Train Epoch: 17 [0/226 (0%)]    Loss: 18.930912
Train Epoch: 17 [64/226 (25%)]  Loss: 16.300846
Train Epoch: 17 [128/226 (50%)] Loss: 21.136532
Train Epoch: 17 [102/226 (75%)] Loss: 19.940079
epoch 17 training loss: 19.077092170715332
epoch 17 validation loss: 18.699552095853367
Train Epoch: 18 [0/226 (0%)]    Loss: 19.193779
Train Epoch: 18 [64/226 (25%)]  Loss: 19.444534
Train Epoch: 18 [128/226 (50%)] Loss: 17.358833
Train Epoch: 18 [102/226 (75%)] Loss: 19.144941
epoch 18 training loss: 18.785521984100342
epoch 18 validation loss: 20.940638615534855
Train Epoch: 19 [0/226 (0%)]    Loss: 18.403282
Train Epoch: 19 [64/226 (25%)]  Loss: 19.547577
Train Epoch: 19 [128/226 (50%)] Loss: 20.563341
Train Epoch: 19 [102/226 (75%)] Loss: 13.532596
epoch 19 training loss: 18.011698961257935
epoch 19 validation loss: 18.51553227351262
Train Epoch: 20 [0/226 (0%)]    Loss: 17.183374
Train Epoch: 20 [64/226 (25%)]  Loss: 20.141644
Train Epoch: 20 [128/226 (50%)] Loss: 19.057522
Train Epoch: 20 [102/226 (75%)] Loss: 12.357776
epoch 20 training loss: 17.185078859329224
epoch 20 validation loss: 21.48157677283654
Train Epoch: 21 [0/226 (0%)]    Loss: 17.199986
Train Epoch: 21 [64/226 (25%)]  Loss: 21.194729
Train Epoch: 21 [128/226 (50%)] Loss: 16.663807
Train Epoch: 21 [102/226 (75%)] Loss: 17.849157
epoch 21 training loss: 18.226919651031494
epoch 21 validation loss: 18.223474355844353
Train Epoch: 22 [0/226 (0%)]    Loss: 20.652115
Train Epoch: 22 [64/226 (25%)]  Loss: 14.307838
Train Epoch: 22 [128/226 (50%)] Loss: 18.288338
Train Epoch: 22 [102/226 (75%)] Loss: 23.672474
epoch 22 training loss: 19.230191230773926
epoch 22 validation loss: 18.65077444223257
Train Epoch: 23 [0/226 (0%)]    Loss: 16.819939
Train Epoch: 23 [64/226 (25%)]  Loss: 17.444836
Train Epoch: 23 [128/226 (50%)] Loss: 17.673868
Train Epoch: 23 [102/226 (75%)] Loss: 20.387896
epoch 23 training loss: 18.081634521484375
epoch 23 validation loss: 18.90660682091346
Train Epoch: 24 [0/226 (0%)]    Loss: 20.353960
Train Epoch: 24 [64/226 (25%)]  Loss: 17.404505
Train Epoch: 24 [128/226 (50%)] Loss: 17.636789
Train Epoch: 24 [102/226 (75%)] Loss: 16.547852
epoch 24 training loss: 17.98577642440796
```

```
epoch 24 validation loss: 20.99890606219952
Train Epoch: 25 [0/226 (0%)]    Loss: 17.858097
Train Epoch: 25 [64/226 (25%)]  Loss: 19.059256
Train Epoch: 25 [128/226 (50%)] Loss: 20.266703
Train Epoch: 25 [102/226 (75%)] Loss: 12.913852
epoch 25 training loss: 17.524476766586304
epoch 25 validation loss: 19.32073270357572
Train Epoch: 26 [0/226 (0%)]    Loss: 17.548508
Train Epoch: 26 [64/226 (25%)]  Loss: 19.725996
Train Epoch: 26 [128/226 (50%)] Loss: 18.696268
Train Epoch: 26 [102/226 (75%)] Loss: 17.362778
epoch 26 training loss: 18.33338737487793
epoch 26 validation loss: 21.722200833834133
Train Epoch: 27 [0/226 (0%)]    Loss: 25.060974
Train Epoch: 27 [64/226 (25%)]  Loss: 23.329670
Train Epoch: 27 [128/226 (50%)] Loss: 19.827322
Train Epoch: 27 [102/226 (75%)] Loss: 22.355362
epoch 27 training loss: 22.64333200454712
epoch 27 validation loss: 23.209383451021633
Train Epoch: 28 [0/226 (0%)]    Loss: 15.743309
Train Epoch: 28 [64/226 (25%)]  Loss: 21.155121
Train Epoch: 28 [128/226 (50%)] Loss: 23.227406
Train Epoch: 28 [102/226 (75%)] Loss: 16.589828
epoch 28 training loss: 19.178915977478027
epoch 28 validation loss: 22.58038330078125
Train Epoch: 29 [0/226 (0%)]    Loss: 18.398947
Train Epoch: 29 [64/226 (25%)]  Loss: 22.236374
Train Epoch: 29 [128/226 (50%)] Loss: 21.519453
Train Epoch: 29 [102/226 (75%)] Loss: 17.229998
epoch 29 training loss: 19.846192836761475
epoch 29 validation loss: 20.85651573768029
Train Epoch: 30 [0/226 (0%)]    Loss: 24.755241
Train Epoch: 30 [64/226 (25%)]  Loss: 17.943787
Train Epoch: 30 [128/226 (50%)] Loss: 15.735841
Train Epoch: 30 [102/226 (75%)] Loss: 23.478064
epoch 30 training loss: 20.478233098983765
epoch 30 validation loss: 20.30492459810697
Train Epoch: 31 [0/226 (0%)]    Loss: 21.364807
Train Epoch: 31 [64/226 (25%)]  Loss: 21.304281
Train Epoch: 31 [128/226 (50%)] Loss: 26.270769
Train Epoch: 31 [102/226 (75%)] Loss: 13.088060
epoch 31 training loss: 20.50697946548462
epoch 31 validation loss: 27.39300537109375
Train Epoch: 32 [0/226 (0%)]    Loss: 23.343134
Train Epoch: 32 [64/226 (25%)]  Loss: 16.888578
Train Epoch: 32 [128/226 (50%)] Loss: 23.155907
Train Epoch: 32 [102/226 (75%)] Loss: 24.749380
epoch 32 training loss: 22.034249782562256
```

```
epoch 32 validation loss: 19.968048095703125
Train Epoch: 33 [0/226 (0%)]    Loss: 16.120749
Train Epoch: 33 [64/226 (25%)]  Loss: 24.155157
Train Epoch: 33 [128/226 (50%)] Loss: 16.861626
Train Epoch: 33 [102/226 (75%)] Loss: 17.471752
epoch 33 training loss: 18.652320861816406
epoch 33 validation loss: 17.773700420673077
Train Epoch: 34 [0/226 (0%)]    Loss: 19.320803
Train Epoch: 34 [64/226 (25%)]  Loss: 21.999252
Train Epoch: 34 [128/226 (50%)] Loss: 15.064029
Train Epoch: 34 [102/226 (75%)] Loss: 15.735632
epoch 34 training loss: 18.0299289226532
epoch 34 validation loss: 18.272613525390625
Train Epoch: 35 [0/226 (0%)]    Loss: 19.761024
Train Epoch: 35 [64/226 (25%)]  Loss: 18.401527
Train Epoch: 35 [128/226 (50%)] Loss: 16.436615
Train Epoch: 35 [102/226 (75%)] Loss: 18.425629
epoch 35 training loss: 18.25619888305664
epoch 35 validation loss: 21.084857647235577
Train Epoch: 36 [0/226 (0%)]    Loss: 17.099131
Train Epoch: 36 [64/226 (25%)]  Loss: 20.742237
Train Epoch: 36 [128/226 (50%)] Loss: 19.155073
Train Epoch: 36 [102/226 (75%)] Loss: 15.789524
epoch 36 training loss: 18.196491241455078
epoch 36 validation loss: 23.685274564302883
Train Epoch: 37 [0/226 (0%)]    Loss: 16.788927
Train Epoch: 37 [64/226 (25%)]  Loss: 27.449089
Train Epoch: 37 [128/226 (50%)] Loss: 17.973572
Train Epoch: 37 [102/226 (75%)] Loss: 24.244341
epoch 37 training loss: 21.61398220062256
epoch 37 validation loss: 19.676881056565506
Train Epoch: 38 [0/226 (0%)]    Loss: 19.467150
Train Epoch: 38 [64/226 (25%)]  Loss: 16.025204
Train Epoch: 38 [128/226 (50%)] Loss: 19.323368
Train Epoch: 38 [102/226 (75%)] Loss: 24.717375
epoch 38 training loss: 19.88327407836914
epoch 38 validation loss: 18.47896517240084
Train Epoch: 39 [0/226 (0%)]    Loss: 18.136311
Train Epoch: 39 [64/226 (25%)]  Loss: 22.594534
Train Epoch: 39 [128/226 (50%)] Loss: 22.150766
Train Epoch: 39 [102/226 (75%)] Loss: 14.764911
epoch 39 training loss: 19.411630392074585
epoch 39 validation loss: 34.0342031625601
Train Epoch: 40 [0/226 (0%)]    Loss: 19.666529
Train Epoch: 40 [64/226 (25%)]  Loss: 40.177250
Train Epoch: 40 [128/226 (50%)] Loss: 18.424906
Train Epoch: 40 [102/226 (75%)] Loss: 25.539867
epoch 40 training loss: 25.95213794708252
```

```
epoch 40 validation loss: 21.40387197641226
Train Epoch: 41 [0/226 (0%)]    Loss: 25.007698
Train Epoch: 41 [64/226 (25%)]  Loss: 19.528822
Train Epoch: 41 [128/226 (50%)] Loss: 26.162992
Train Epoch: 41 [102/226 (75%)] Loss: 24.425613
epoch 41 training loss: 23.78128147125244
epoch 41 validation loss: 18.126678466796875
Train Epoch: 42 [0/226 (0%)]    Loss: 18.275412
Train Epoch: 42 [64/226 (25%)]  Loss: 37.806702
Train Epoch: 42 [128/226 (50%)] Loss: 19.802427
Train Epoch: 42 [102/226 (75%)] Loss: 28.518679
epoch 42 training loss: 26.100804805755615
epoch 42 validation loss: 24.845008263221153
Train Epoch: 43 [0/226 (0%)]    Loss: 29.054794
Train Epoch: 43 [64/226 (25%)]  Loss: 22.961458
Train Epoch: 43 [128/226 (50%)] Loss: 19.849289
Train Epoch: 43 [102/226 (75%)] Loss: 17.243702
epoch 43 training loss: 22.277310848236084
epoch 43 validation loss: 25.212740384615383
Train Epoch: 44 [0/226 (0%)]    Loss: 19.215229
Train Epoch: 44 [64/226 (25%)]  Loss: 24.224560
Train Epoch: 44 [128/226 (50%)] Loss: 17.454460
Train Epoch: 44 [102/226 (75%)] Loss: 22.325037
epoch 44 training loss: 20.804821491241455
epoch 44 validation loss: 17.368219228891228
Train Epoch: 45 [0/226 (0%)]    Loss: 15.572681
Train Epoch: 45 [64/226 (25%)]  Loss: 23.434982
Train Epoch: 45 [128/226 (50%)] Loss: 14.484062
Train Epoch: 45 [102/226 (75%)] Loss: 22.477365
epoch 45 training loss: 18.99227285385132
epoch 45 validation loss: 17.27588125375601
Train Epoch: 46 [0/226 (0%)]    Loss: 17.057096
Train Epoch: 46 [64/226 (25%)]  Loss: 14.759440
Train Epoch: 46 [128/226 (50%)] Loss: 21.324604
Train Epoch: 46 [102/226 (75%)] Loss: 19.819223
epoch 46 training loss: 18.24009108543396
epoch 46 validation loss: 21.769812950721153
Train Epoch: 47 [0/226 (0%)]    Loss: 20.560619
Train Epoch: 47 [64/226 (25%)]  Loss: 18.297558
Train Epoch: 47 [128/226 (50%)] Loss: 23.810825
Train Epoch: 47 [102/226 (75%)] Loss: 15.435053
epoch 47 training loss: 19.52601385116577
epoch 47 validation loss: 21.67011437049279
Train Epoch: 48 [0/226 (0%)]    Loss: 20.548138
Train Epoch: 48 [64/226 (25%)]  Loss: 15.620457
Train Epoch: 48 [128/226 (50%)] Loss: 19.592726
Train Epoch: 48 [102/226 (75%)] Loss: 16.831966
epoch 48 training loss: 18.148321628570557
```

```
epoch 48 validation loss: 17.55987079326923
Train Epoch: 49 [0/226 (0%)]    Loss: 19.234060
Train Epoch: 49 [64/226 (25%)]  Loss: 17.107624
Train Epoch: 49 [128/226 (50%)] Loss: 15.571829
Train Epoch: 49 [102/226 (75%)] Loss: 17.757883
epoch 49 training loss: 17.41784906387329
epoch 49 validation loss: 19.306349534254807
Train Epoch: 50 [0/226 (0%)]    Loss: 18.032560
Train Epoch: 50 [64/226 (25%)]  Loss: 15.419507
Train Epoch: 50 [128/226 (50%)] Loss: 17.053217
Train Epoch: 50 [102/226 (75%)] Loss: 21.950911
epoch 50 training loss: 18.114048719406128
epoch 50 validation loss: 21.6327397273137
Train Epoch: 51 [0/226 (0%)]    Loss: 21.231552
Train Epoch: 51 [64/226 (25%)]  Loss: 16.274305
Train Epoch: 51 [128/226 (50%)] Loss: 15.278790
Train Epoch: 51 [102/226 (75%)] Loss: 26.681717
epoch 51 training loss: 19.866590976715088
epoch 51 validation loss: 18.051175631009617
Train Epoch: 52 [0/226 (0%)]    Loss: 14.597397
Train Epoch: 52 [64/226 (25%)]  Loss: 26.404018
Train Epoch: 52 [128/226 (50%)] Loss: 11.334438
Train Epoch: 52 [102/226 (75%)] Loss: 26.849089
epoch 52 training loss: 19.79623556137085
epoch 52 validation loss: 17.693087064302883
Train Epoch: 53 [0/226 (0%)]    Loss: 18.393604
Train Epoch: 53 [64/226 (25%)]  Loss: 18.503284
Train Epoch: 53 [128/226 (50%)] Loss: 14.868428
Train Epoch: 53 [102/226 (75%)] Loss: 16.611359
epoch 53 training loss: 17.09416890144348
epoch 53 validation loss: 17.6173095703125
Train Epoch: 54 [0/226 (0%)]    Loss: 14.958791
Train Epoch: 54 [64/226 (25%)]  Loss: 22.938211
Train Epoch: 54 [128/226 (50%)] Loss: 17.025208
Train Epoch: 54 [102/226 (75%)] Loss: 22.779882
epoch 54 training loss: 19.425523042678833
epoch 54 validation loss: 26.87799072265625
Train Epoch: 55 [0/226 (0%)]    Loss: 22.968332
Train Epoch: 55 [64/226 (25%)]  Loss: 15.613537
Train Epoch: 55 [128/226 (50%)] Loss: 25.916752
Train Epoch: 55 [102/226 (75%)] Loss: 25.889090
epoch 55 training loss: 22.596927642822266
epoch 55 validation loss: 23.77826162484976
Train Epoch: 56 [0/226 (0%)]    Loss: 24.043446
Train Epoch: 56 [64/226 (25%)]  Loss: 21.750481
Train Epoch: 56 [128/226 (50%)] Loss: 21.691734
Train Epoch: 56 [102/226 (75%)] Loss: 20.506805
epoch 56 training loss: 21.998116493225098
```

```
epoch 56 validation loss: 29.61139855018029
Train Epoch: 57 [0/226 (0%)]    Loss: 42.164005
Train Epoch: 57 [64/226 (25%)]  Loss: 15.701812
Train Epoch: 57 [128/226 (50%)] Loss: 25.219799
Train Epoch: 57 [102/226 (75%)] Loss: 31.212166
epoch 57 training loss: 28.574445486068726
epoch 57 validation loss: 19.999187762920673
Train Epoch: 58 [0/226 (0%)]    Loss: 19.065077
Train Epoch: 58 [64/226 (25%)]  Loss: 25.602255
Train Epoch: 58 [128/226 (50%)] Loss: 22.461138
Train Epoch: 58 [102/226 (75%)] Loss: 13.055816
epoch 58 training loss: 20.04607129096985
epoch 58 validation loss: 30.410428560697117
Train Epoch: 59 [0/226 (0%)]    Loss: 22.128490
Train Epoch: 59 [64/226 (25%)]  Loss: 33.147453
Train Epoch: 59 [128/226 (50%)] Loss: 17.127132
Train Epoch: 59 [102/226 (75%)] Loss: 29.710821
epoch 59 training loss: 25.5284743309021
epoch 59 validation loss: 18.727962787334736
Train Epoch: 60 [0/226 (0%)]    Loss: 23.815207
Train Epoch: 60 [64/226 (25%)]  Loss: 13.918279
Train Epoch: 60 [128/226 (50%)] Loss: 24.291220
Train Epoch: 60 [102/226 (75%)] Loss: 19.090744
epoch 60 training loss: 20.278862237930298
epoch 60 validation loss: 17.24773230919471
Train Epoch: 61 [0/226 (0%)]    Loss: 18.670824
Train Epoch: 61 [64/226 (25%)]  Loss: 24.765026
Train Epoch: 61 [128/226 (50%)] Loss: 19.276672
Train Epoch: 61 [102/226 (75%)] Loss: 24.432846
epoch 61 training loss: 21.78634214401245
epoch 61 validation loss: 32.56958242563101
Train Epoch: 62 [0/226 (0%)]    Loss: 30.126207
Train Epoch: 62 [64/226 (25%)]  Loss: 23.047781
Train Epoch: 62 [128/226 (50%)] Loss: 24.804245
Train Epoch: 62 [102/226 (75%)] Loss: 19.290525
epoch 62 training loss: 24.317189693450928
epoch 62 validation loss: 19.61239741398738
Train Epoch: 63 [0/226 (0%)]    Loss: 18.657673
Train Epoch: 63 [64/226 (25%)]  Loss: 25.003096
Train Epoch: 63 [128/226 (50%)] Loss: 17.537830
Train Epoch: 63 [102/226 (75%)] Loss: 17.221016
epoch 63 training loss: 19.60490369796753
epoch 63 validation loss: 20.645040658804085
Train Epoch: 64 [0/226 (0%)]    Loss: 23.134041
Train Epoch: 64 [64/226 (25%)]  Loss: 17.908333
Train Epoch: 64 [128/226 (50%)] Loss: 18.299147
Train Epoch: 64 [102/226 (75%)] Loss: 17.289364
epoch 64 training loss: 19.157721042633057
```

```
epoch 64 validation loss: 23.43372521033654
Train Epoch: 65 [0/226 (0%)]    Loss: 22.569786
Train Epoch: 65 [64/226 (25%)]  Loss: 15.493501
Train Epoch: 65 [128/226 (50%)] Loss: 22.313198
Train Epoch: 65 [102/226 (75%)] Loss: 20.464714
epoch 65 training loss: 20.210299730300903
epoch 65 validation loss: 22.846717247596153
Train Epoch: 66 [0/226 (0%)]    Loss: 21.388763
Train Epoch: 66 [64/226 (25%)]  Loss: 18.639383
Train Epoch: 66 [128/226 (50%)] Loss: 16.805304
Train Epoch: 66 [102/226 (75%)] Loss: 17.212540
epoch 66 training loss: 18.511497497558594
epoch 66 validation loss: 19.814483642578125
Train Epoch: 67 [0/226 (0%)]    Loss: 26.163090
Train Epoch: 67 [64/226 (25%)]  Loss: 15.576175
Train Epoch: 67 [128/226 (50%)] Loss: 21.717102
Train Epoch: 67 [102/226 (75%)] Loss: 19.998188
epoch 67 training loss: 20.863638639450073
epoch 67 validation loss: 17.762404221754807
Train Epoch: 68 [0/226 (0%)]    Loss: 16.363428
Train Epoch: 68 [64/226 (25%)]  Loss: 17.417063
Train Epoch: 68 [128/226 (50%)] Loss: 24.225122
Train Epoch: 68 [102/226 (75%)] Loss: 13.549788
epoch 68 training loss: 17.888850212097168
epoch 68 validation loss: 25.14800555889423
Train Epoch: 69 [0/226 (0%)]    Loss: 24.288324
Train Epoch: 69 [64/226 (25%)]  Loss: 16.836767
Train Epoch: 69 [128/226 (50%)] Loss: 18.475279
Train Epoch: 69 [102/226 (75%)] Loss: 22.214346
epoch 69 training loss: 20.453679084777832
epoch 69 validation loss: 18.16720463679387
Train Epoch: 70 [0/226 (0%)]    Loss: 19.013067
Train Epoch: 70 [64/226 (25%)]  Loss: 17.296921
Train Epoch: 70 [128/226 (50%)] Loss: 15.827392
Train Epoch: 70 [102/226 (75%)] Loss: 25.978165
epoch 70 training loss: 19.528886079788208
epoch 70 validation loss: 18.37356684758113
Train Epoch: 71 [0/226 (0%)]    Loss: 17.337132
Train Epoch: 71 [64/226 (25%)]  Loss: 23.918913
Train Epoch: 71 [128/226 (50%)] Loss: 14.417519
Train Epoch: 71 [102/226 (75%)] Loss: 16.824202
epoch 71 training loss: 18.124441146850586
epoch 71 validation loss: 17.752838134765625
Train Epoch: 72 [0/226 (0%)]    Loss: 20.140865
Train Epoch: 72 [64/226 (25%)]  Loss: 18.887321
Train Epoch: 72 [128/226 (50%)] Loss: 14.108324
Train Epoch: 72 [102/226 (75%)] Loss: 13.786165
epoch 72 training loss: 16.730669021606445
```

```
epoch 72 validation loss: 20.34656935471755
Train Epoch: 73 [0/226 (0%)]    Loss: 17.432943
Train Epoch: 73 [64/226 (25%)]  Loss: 17.157284
Train Epoch: 73 [128/226 (50%)] Loss: 14.757550
Train Epoch: 73 [102/226 (75%)] Loss: 13.101101
epoch 73 training loss: 15.61221957206726
epoch 73 validation loss: 17.62223698542668
Train Epoch: 74 [0/226 (0%)]    Loss: 18.643028
Train Epoch: 74 [64/226 (25%)]  Loss: 15.744242
Train Epoch: 74 [128/226 (50%)] Loss: 16.741968
Train Epoch: 74 [102/226 (75%)] Loss: 18.184265
epoch 74 training loss: 17.328375816345215
epoch 74 validation loss: 16.95874727689303
Train Epoch: 75 [0/226 (0%)]    Loss: 17.208561
Train Epoch: 75 [64/226 (25%)]  Loss: 19.913343
Train Epoch: 75 [128/226 (50%)] Loss: 14.519900
Train Epoch: 75 [102/226 (75%)] Loss: 11.173885
epoch 75 training loss: 15.703922510147095
epoch 75 validation loss: 20.169170673076923
Train Epoch: 76 [0/226 (0%)]    Loss: 15.721628
Train Epoch: 76 [64/226 (25%)]  Loss: 21.367245
Train Epoch: 76 [128/226 (50%)] Loss: 15.074806
Train Epoch: 76 [102/226 (75%)] Loss: 23.232773
epoch 76 training loss: 18.84911298751831
epoch 76 validation loss: 16.596265352689304
Train Epoch: 77 [0/226 (0%)]    Loss: 15.602701
Train Epoch: 77 [64/226 (25%)]  Loss: 16.827318
Train Epoch: 77 [128/226 (50%)] Loss: 20.026108
Train Epoch: 77 [102/226 (75%)] Loss: 15.537221
epoch 77 training loss: 16.998337030410767
epoch 77 validation loss: 16.9235346867488
Train Epoch: 78 [0/226 (0%)]    Loss: 15.130124
Train Epoch: 78 [64/226 (25%)]  Loss: 18.413574
Train Epoch: 78 [128/226 (50%)] Loss: 15.137374
Train Epoch: 78 [102/226 (75%)] Loss: 15.717416
epoch 78 training loss: 16.099622011184692
epoch 78 validation loss: 18.68586848332332
Train Epoch: 79 [0/226 (0%)]    Loss: 16.869553
Train Epoch: 79 [64/226 (25%)]  Loss: 13.905661
Train Epoch: 79 [128/226 (50%)] Loss: 17.502373
Train Epoch: 79 [102/226 (75%)] Loss: 18.219419
epoch 79 training loss: 16.62425136566162
epoch 79 validation loss: 16.505010751577522
Train Epoch: 80 [0/226 (0%)]    Loss: 15.987420
Train Epoch: 80 [64/226 (25%)]  Loss: 17.659853
Train Epoch: 80 [128/226 (50%)] Loss: 15.353962
Train Epoch: 80 [102/226 (75%)] Loss: 21.203863
epoch 80 training loss: 17.55127453804016
```

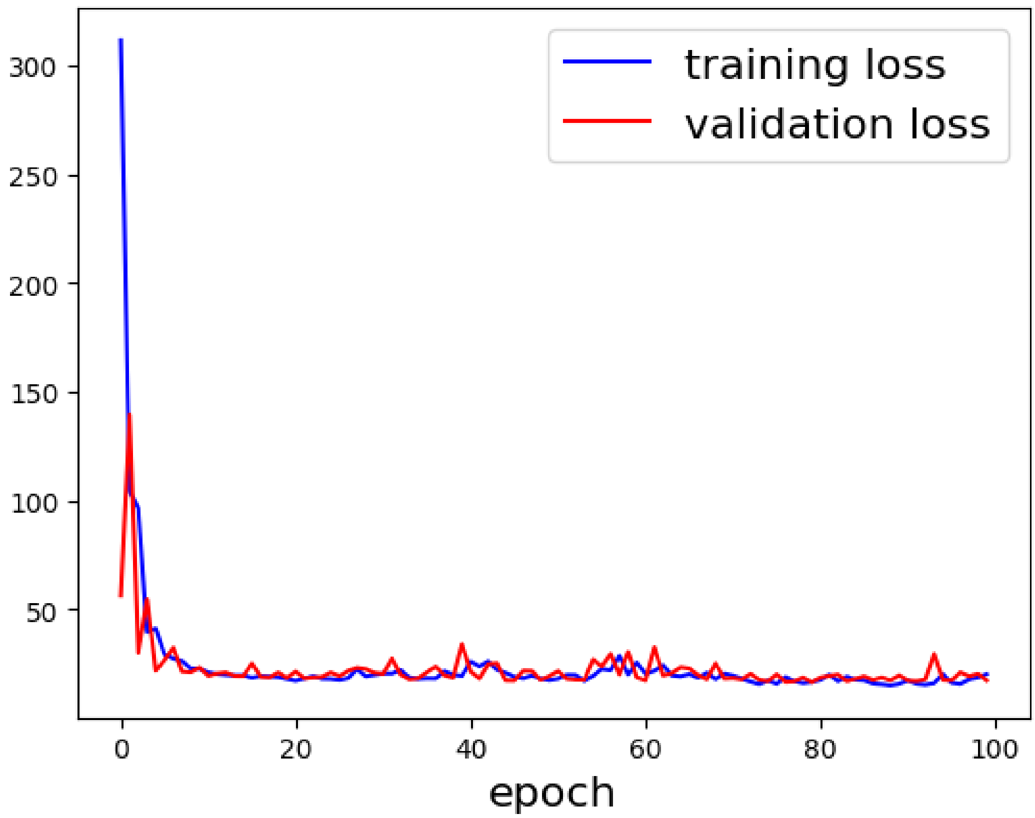
```
epoch 80 validation loss: 18.647970346304085
Train Epoch: 81 [0/226 (0%)]    Loss: 19.032248
Train Epoch: 81 [64/226 (25%)]  Loss: 19.928110
Train Epoch: 81 [128/226 (50%)] Loss: 15.116875
Train Epoch: 81 [102/226 (75%)] Loss: 25.651667
epoch 81 training loss: 19.9322247505188
epoch 81 validation loss: 19.37487323467548
Train Epoch: 82 [0/226 (0%)]    Loss: 16.481026
Train Epoch: 82 [64/226 (25%)]  Loss: 16.388685
Train Epoch: 82 [128/226 (50%)] Loss: 15.070370
Train Epoch: 82 [102/226 (75%)] Loss: 19.900524
epoch 82 training loss: 16.960151195526123
epoch 82 validation loss: 20.03504356971154
Train Epoch: 83 [0/226 (0%)]    Loss: 16.324112
Train Epoch: 83 [64/226 (25%)]  Loss: 16.532368
Train Epoch: 83 [128/226 (50%)] Loss: 22.278671
Train Epoch: 83 [102/226 (75%)] Loss: 21.193233
epoch 83 training loss: 19.082096099853516
epoch 83 validation loss: 16.98590087890625
Train Epoch: 84 [0/226 (0%)]    Loss: 15.468060
Train Epoch: 84 [64/226 (25%)]  Loss: 18.602020
Train Epoch: 84 [128/226 (50%)] Loss: 17.054569
Train Epoch: 84 [102/226 (75%)] Loss: 19.377686
epoch 84 training loss: 17.62558364868164
epoch 84 validation loss: 18.081873967097355
Train Epoch: 85 [0/226 (0%)]    Loss: 23.111620
Train Epoch: 85 [64/226 (25%)]  Loss: 13.773888
Train Epoch: 85 [128/226 (50%)] Loss: 16.320282
Train Epoch: 85 [102/226 (75%)] Loss: 16.402201
epoch 85 training loss: 17.401997566223145
epoch 85 validation loss: 19.193841787484978
Train Epoch: 86 [0/226 (0%)]    Loss: 17.507809
Train Epoch: 86 [64/226 (25%)]  Loss: 16.395199
Train Epoch: 86 [128/226 (50%)] Loss: 15.236087
Train Epoch: 86 [102/226 (75%)] Loss: 14.301262
epoch 86 training loss: 15.86008906364441
epoch 86 validation loss: 17.491935143103966
Train Epoch: 87 [0/226 (0%)]    Loss: 19.686155
Train Epoch: 87 [64/226 (25%)]  Loss: 13.862446
Train Epoch: 87 [128/226 (50%)] Loss: 14.819418
Train Epoch: 87 [102/226 (75%)] Loss: 13.789185
epoch 87 training loss: 15.539300918579102
epoch 87 validation loss: 18.667947622445915
Train Epoch: 88 [0/226 (0%)]    Loss: 19.789534
Train Epoch: 88 [64/226 (25%)]  Loss: 14.899501
Train Epoch: 88 [128/226 (50%)] Loss: 13.937860
Train Epoch: 88 [102/226 (75%)] Loss: 11.510489
epoch 88 training loss: 15.034345626831055
```

```
epoch 88 validation loss: 17.34429931640625
Train Epoch: 89 [0/226 (0%)]    Loss: 15.819696
Train Epoch: 89 [64/226 (25%)]  Loss: 14.120276
Train Epoch: 89 [128/226 (50%)] Loss: 20.105145
Train Epoch: 89 [102/226 (75%)] Loss: 12.864442
epoch 89 training loss: 15.727389812469482
epoch 89 validation loss: 19.67983656663161
Train Epoch: 90 [0/226 (0%)]    Loss: 16.957342
Train Epoch: 90 [64/226 (25%)]  Loss: 13.656474
Train Epoch: 90 [128/226 (50%)] Loss: 17.234324
Train Epoch: 90 [102/226 (75%)] Loss: 21.543058
epoch 90 training loss: 17.34779953956604
epoch 90 validation loss: 17.26756873497596
Train Epoch: 91 [0/226 (0%)]    Loss: 14.890552
Train Epoch: 91 [64/226 (25%)]  Loss: 16.147953
Train Epoch: 91 [128/226 (50%)] Loss: 13.954403
Train Epoch: 91 [102/226 (75%)] Loss: 18.077536
epoch 91 training loss: 15.767610788345337
epoch 91 validation loss: 17.038211529071514
Train Epoch: 92 [0/226 (0%)]    Loss: 15.026597
Train Epoch: 92 [64/226 (25%)]  Loss: 18.412588
Train Epoch: 92 [128/226 (50%)] Loss: 12.790339
Train Epoch: 92 [102/226 (75%)] Loss: 15.145915
epoch 92 training loss: 15.343859910964966
epoch 92 validation loss: 17.845716036283054
Train Epoch: 93 [0/226 (0%)]    Loss: 15.498976
Train Epoch: 93 [64/226 (25%)]  Loss: 19.713676
Train Epoch: 93 [128/226 (50%)] Loss: 14.782675
Train Epoch: 93 [102/226 (75%)] Loss: 14.290501
epoch 93 training loss: 16.071456909179688
epoch 93 validation loss: 29.618072509765625
Train Epoch: 94 [0/226 (0%)]    Loss: 28.665785
Train Epoch: 94 [64/226 (25%)]  Loss: 16.521168
Train Epoch: 94 [128/226 (50%)] Loss: 20.277328
Train Epoch: 94 [102/226 (75%)] Loss: 15.268829
epoch 94 training loss: 20.18327760696411
epoch 94 validation loss: 17.5186039851262
Train Epoch: 95 [0/226 (0%)]    Loss: 15.149218
Train Epoch: 95 [64/226 (25%)]  Loss: 21.774418
Train Epoch: 95 [128/226 (50%)] Loss: 14.715351
Train Epoch: 95 [102/226 (75%)] Loss: 13.352221
epoch 95 training loss: 16.247802019119263
epoch 95 validation loss: 17.40655282827524
Train Epoch: 96 [0/226 (0%)]    Loss: 13.590445
Train Epoch: 96 [64/226 (25%)]  Loss: 20.739527
Train Epoch: 96 [128/226 (50%)] Loss: 15.560705
Train Epoch: 96 [102/226 (75%)] Loss: 12.862264
epoch 96 training loss: 15.68823504447937
```

```
epoch 96 validation loss: 21.029883751502403
Train Epoch: 97 [0/226 (0%)]    Loss: 19.062233
Train Epoch: 97 [64/226 (25%)]  Loss: 14.018667
Train Epoch: 97 [128/226 (50%)] Loss: 13.828022
Train Epoch: 97 [102/226 (75%)] Loss: 24.399380
epoch 97 training loss: 17.827075481414795
epoch 97 validation loss: 19.217987060546875
Train Epoch: 98 [0/226 (0%)]    Loss: 17.717041
Train Epoch: 98 [64/226 (25%)]  Loss: 25.305286
Train Epoch: 98 [128/226 (50%)] Loss: 12.244699
Train Epoch: 98 [102/226 (75%)] Loss: 19.209782
epoch 98 training loss: 18.619202136993408
epoch 98 validation loss: 20.370143010066105
Train Epoch: 99 [0/226 (0%)]    Loss: 19.359550
Train Epoch: 99 [64/226 (25%)]  Loss: 17.510099
Train Epoch: 99 [128/226 (50%)] Loss: 20.818611
Train Epoch: 99 [102/226 (75%)] Loss: 23.117645
epoch 99 training loss: 20.201476573944092
epoch 99 validation loss: 17.349280724158653
```

[ ]:
```
fig, ax = plt.subplots()
ax.plot(np.arange(0,len(loss_train_list)), loss_train_list, '-b',␣
 ↪label='training loss')
ax.plot(np.arange(0,len(loss_val_list)), loss_val_list, '-r', label='validation␣
 ↪loss')
ax.set_xlabel('epoch',fontsize=16)
ax.legend(fontsize=16)
```

[ ]:
```
<matplotlib.legend.Legend at 0x7f7c7a430760>
```

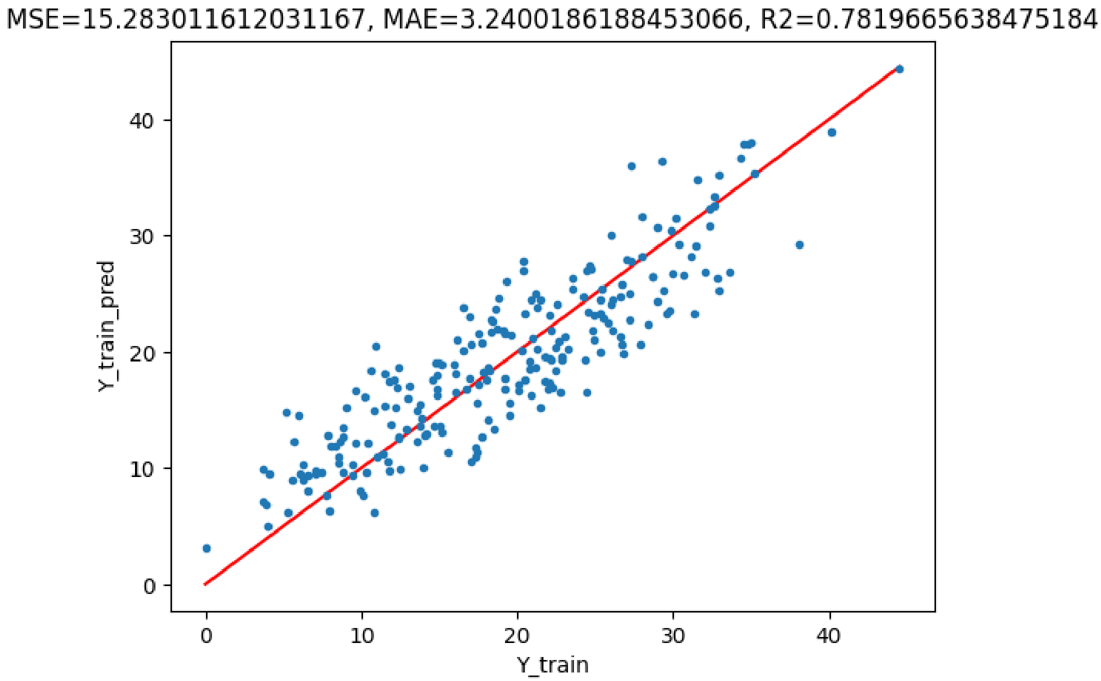


15 Training results:

```
[ ]: X_train_t=torch.tensor(X_train_s, dtype=torch.float32) #convert numpy array to␣
     ↪torch tensor
     #the input to model should be torch tensor
     Y_train_pred_t=model(X_train_t.to(device))
     Y_train_pred=Y_train_pred_t.detach().cpu().numpy() #convert torch tensor to␣
     ↪numpy array
```

```
[ ]: MSE = np.mean((Y_train - Y_train_pred)**2)
     MAE = np.mean(np.abs(Y_train - Y_train_pred))
     R2 = r2_score(Y_train, Y_train_pred)
     #
     ymax=np.max([Y_train.max(), Y_train_pred.max()])
     plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
     plt.plot(Y_train, Y_train_pred, '.')
     plt.xlabel('Y_train')
     plt.ylabel('Y_train_pred')
     plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[ ]: Text(0.5, 1.0, 'MSE=15.283011612031167, MAE=3.2400186188453066, R2=0.7819665638475184')

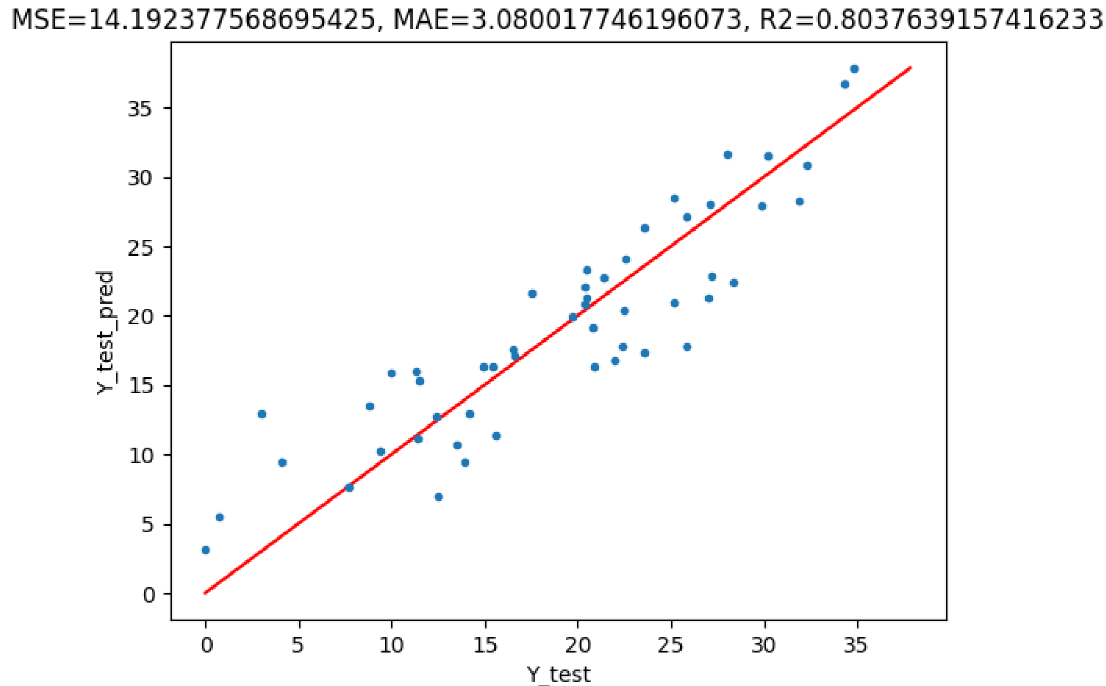


16 Testing results:

```
[ ]: X_test_t=torch.tensor(X_test_s, dtype=torch.float32) #convert numpy array to torch tensor
     #the input to model should be torch tensor
     Y_test_pred_t=model(X_test_t.to(device))
     Y_test_pred=Y_test_pred_t.detach().cpu().numpy() #convert torch tensor to numpy array
```

```
[ ]: MSE = np.mean((Y_test - Y_test_pred)**2)
     MAE = np.mean(np.abs(Y_test - Y_test_pred))
     R2 = r2_score(Y_test, Y_test_pred)
     #
     ymax=np.max([Y_test.max(), Y_test_pred.max()])
     plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
     plt.plot(Y_test, Y_test_pred, '.')
     plt.xlabel('Y_test')
     plt.ylabel('Y_test_pred')
     plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[ ]: Text(0.5, 1.0, 'MSE=14.192377568695425, MAE=3.080017746196073,
R2=0.8037639157416233')



17 Conclusion part 3

The Neural Networks shown slightly better performance than the average. The nonlinear network has been the top performer so far, achieving an R2 score of 0.8 on the test set.

---

18 Part 4

18.1 Using Auto-PyTorch

Auto-PyTorch jointly and robustly optimizes the network architecture and the training hyperparameters to enable fully automated deep learning (AutoDL).

Auto-PyTorch is mainly developed to support tabular data (classification, regression) and time series data (forecasting).

[ ]:
```
!pip install torch

!pip install git+https://github.com/shukon/HpBandSter.git
!pip install git+https://github.com/automl/Auto-PyTorch.git
```

```
[ ]: !pip install -U scikit-learn
```

```
[ ]: ! sudo apt install msttcorefonts -qq
     ! rm ~/.cache/matplotlib -rf
```

```
[ ]: import matplotlib.pyplot as plt
     import numpy as np
     import torch
     from sklearn.model_selection import train_test_split
     import pandas as pd
     import sklearn.model_selection
     import sklearn.datasets
     import sklearn.metrics
```

```
[ ]: from autoPyTorch.api.tabular_regression import TabularRegressionTask
```

19 Data Preparation

```
[ ]: X = pd.read_csv('X.csv')
     Y = pd.read_csv('Y.csv')
     sklearn.model_selectionX = pd.read_csv('X.csv')
     X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
      ↪random_state=0)
```

```
[ ]: from sklearn.preprocessing import StandardScaler

     X_train=X_train.values

     Y_train=Y_train.values

     X_test=X_test.values

     Y_test=Y_test.values

     scaler=StandardScaler()

     scaler.fit(X_train)

     #normalize the features in the training set
     X_train_s = scaler.transform(X_train)
     #normalize the features in the test set
     X_test_s = scaler.transform(X_test)
```

20 Looking got the best model

[ ]:
```
api = TabularRegressionTask()
```

[ ]:
```
api.search(
    X_train=X_train_s,
    y_train=Y_train,
    X_test=X_test_s.copy(),
    y_test=Y_test.copy(),
    optimize_metric='r2',
    total_walltime_limit=300,
    func_eval_time_limit_secs=50,
    dataset_name="Bodyfat"
)
```

```
/usr/local/lib/python3.10/dist-packages/autoPyTorch/pipeline/components/preproce
ssing/tabular_preprocessing/feature_preprocessing/Nystroem.py:130: UserWarning:
Given choices for `score_func` are not compatible with the dataset. Updating
choices to ['poly', 'rbf', 'sigmoid', 'cosine']
  warnings.warn(f"Given choices for `score_func` are not compatible with the
dataset. "
```

```
[ERROR] [2023-05-04 23:40:35,326:Client-AutoPyTorch:Bodyfat:1] Prediction for
lgb failed with run state StatusType.CRASHED.
Additional info:
traceback: Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/autoPyTorch/evaluation/tae.py",
line 61, in fit_predict_try_except_decorator
    ta(queue=queue, **kwargs)
  File "/usr/local/lib/python3.10/dist-
packages/autoPyTorch/evaluation/train_evaluator.py", line 512, in
eval_train_function
    evaluator.fit_predict_and_loss()
  File "/usr/local/lib/python3.10/dist-
packages/autoPyTorch/evaluation/train_evaluator.py", line 186, in
fit_predict_and_loss
    y_train_pred, y_opt_pred, y_valid_pred, y_test_pred =
self._fit_and_predict(pipeline, split_id,
  File "/usr/local/lib/python3.10/dist-
packages/autoPyTorch/evaluation/train_evaluator.py", line 364, in
_fit_and_predict
    fit_and_suppress_warnings(self.logger, pipeline, X, y)
  File "/usr/local/lib/python3.10/dist-
packages/autoPyTorch/evaluation/abstract_evaluator.py", line 339, in
fit_and_suppress_warnings
    pipeline.fit(X, y)
  File "/usr/local/lib/python3.10/dist-
packages/autoPyTorch/evaluation/abstract_evaluator.py", line 181, in fit
```

```
    return self.pipeline.fit(X, y)
  File "/usr/local/lib/python3.10/dist-
packages/autoPyTorch/pipeline/base_pipeline.py", line 155, in fit
  File "/usr/local/lib/python3.10/dist-
packages/autoPyTorch/pipeline/base_pipeline.py", line 174, in fit_estimator
    self._final_estimator.fit(X, y, **fit_params)
  File "/usr/local/lib/python3.10/dist-
packages/autoPyTorch/pipeline/components/base_choice.py", line 217, in fit
    return self.choice.fit(X, y)
  File "/usr/local/lib/python3.10/dist-
packages/autoPyTorch/pipeline/components/setup/traditional_ml/base_model.py",
line 98, in fit
    self.fit_output = self.model.fit(X['X_train'][X['train_indices']],
X['y_train'][X['train_indices']],
  File "/usr/local/lib/python3.10/dist-packages/autoPyTorch/pipeline/components/
setup/traditional_ml/traditional_learner/base_traditional_learner.py", line 184,
in fit
    self._fit(X_train, y_train, X_val, y_val)
  File "/usr/local/lib/python3.10/dist-packages/autoPyTorch/pipeline/components/
setup/traditional_ml/traditional_learner/learners.py", line 69, in _fit
    self.model.fit(X_train, y_train, eval_set=eval_set)
  File "/usr/local/lib/python3.10/dist-packages/lightgbm/sklearn.py", line 895,
in fit
    super().fit(X, y, sample_weight=sample_weight, init_score=init_score,
  File "/usr/local/lib/python3.10/dist-packages/lightgbm/sklearn.py", line 748,
in fit
    self._Booster = train(
  File "/usr/local/lib/python3.10/dist-packages/lightgbm/engine.py", line 292,
in train
    booster.update(fobj=fobj)
  File "/usr/local/lib/python3.10/dist-packages/lightgbm/basic.py", line 3021,
in update
    _safe_call(_LIB.LGBM_BoosterUpdateOneIter(
  File "/usr/local/lib/python3.10/dist-packages/lightgbm/basic.py", line 125, in
_safe_call
    raise LightGBMError(_LIB.LGBM_GetLastError().decode('utf-8'))
lightgbm.basic.LightGBMError: std::bad_alloc

error: LightGBMError('std::bad_alloc')
configuration_origin: traditional
[ERROR] [2023-05-04 23:40:36,945:Client-AutoPyTorch:Bodyfat:1] Prediction for
catboost failed with run state StatusType.CRASHED,
because the provided memory limits were too tight.
Please increase the 'ml_memory_limit' and try again.
If you still get the problem, please open an issue
and paste the additional info.
Additional info:
error: Result queue is empty
```

```
exit_status: <class 'pynisher.limit_function_call.AnythingException'>
subprocess_stdout:
subprocess_stderr:
exitcode: -6
configuration_origin: traditional
```

```
/usr/local/lib/python3.10/dist-
packages/smac/intensification/parallel_scheduling.py:154: UserWarning: Hyperband
is executed with 1 workers only. Consider to use pynisher to use all available
workers.
  warnings.warn(
```

[ ]: <autoPyTorch.api.tabular_regression.TabularRegressionTask at 0x7fe3aae7f040>

[ ]:
```
# Print statistics from search
print(api.sprint_statistics())
```

```
autoPyTorch results:
        Dataset name: Bodyfat
        Optimisation Metric: r2
        Best validation score: 0.7067779040639525
        Number of target algorithm runs: 36
        Number of successful target algorithm runs: 15
        Number of crashed target algorithm runs: 19
        Number of target algorithms that exceeded the time limit: 1
        Number of target algorithms that exceeded the memory limit: 1
```

Refiting the models on the full dataset:

[ ]:
```
api.refit(
    X_train=X_train_s,
    y_train=Y_train,
    X_test=X_test_s,
    y_test=Y_test,
    dataset_name="BodyFat",
    total_walltime_limit=500,
    run_time_limit_secs=50
)
```

```
/usr/local/lib/python3.10/dist-packages/autoPyTorch/pipeline/components/preproce
ssing/tabular_preprocessing/feature_preprocessing/Nystroem.py:130: UserWarning:
Given choices for `score_func` are not compatible with the dataset. Updating
choices to ['poly', 'rbf', 'sigmoid', 'cosine']
  warnings.warn(f"Given choices for `score_func` are not compatible with the
dataset. "
```

```
[WARNING] [2023-05-04 23:45:57,810:Client-AutoPyTorch:RefitLogger:1] Something
went wrong while processing the results of extra_trees.with additional_info:
{'opt_loss': {'r2': 0.2538095264318905}, 'duration': 2.2709174156188965,
```

```
'num_run': 34, 'train_loss': {'r2': 4.3298697960381105e-15}, 'test_loss': {'r2':
0.2538095264318905}, 'configuration': 'extra_trees', 'budget': 50.0,
'configuration_origin': 'traditional'} and status_type: StatusType.SUCCESS.
Refer to the log file for more information.
Skipping for now.
[WARNING] [2023-05-04 23:46:00,450:Client-AutoPyTorch:RefitLogger:1] Something
went wrong while processing the results of knn.with additional_info:
{'opt_loss': {'r2': 0.3486898881352054}, 'duration': 1.5086331367492676,
'num_run': 35, 'train_loss': {'r2': 0.25564214668631835}, 'test_loss': {'r2':
0.3486898881352054}, 'configuration': 'knn', 'budget': 50.0,
'configuration_origin': 'traditional'} and status_type: StatusType.SUCCESS.
Refer to the log file for more information.
Skipping for now.
```

[ ]: <autoPyTorch.api.tabular_regression.TabularRegressionTask at 0x7fe3aae7f040>

[ ]:
```
Y_test_pred = api.predict(X_test_s)
score = api.score(Y_test_pred, Y_test)
print(score)

# Print the final ensemble built by AutoPyTorch
print(api.show_models())
```

```
{'r2': 0.7603824543757912}
|    | Preprocessing    | Estimator    |   Weight |
|---:|:-----------------|:-------------|---------:|
|  0 | None             | ETLearner    |     0.92 |
|  1 | None             | KNNLearner   |     0.08 |
```

[ ]:
```
Y_train_pred=api.predict(X_train_s)
```

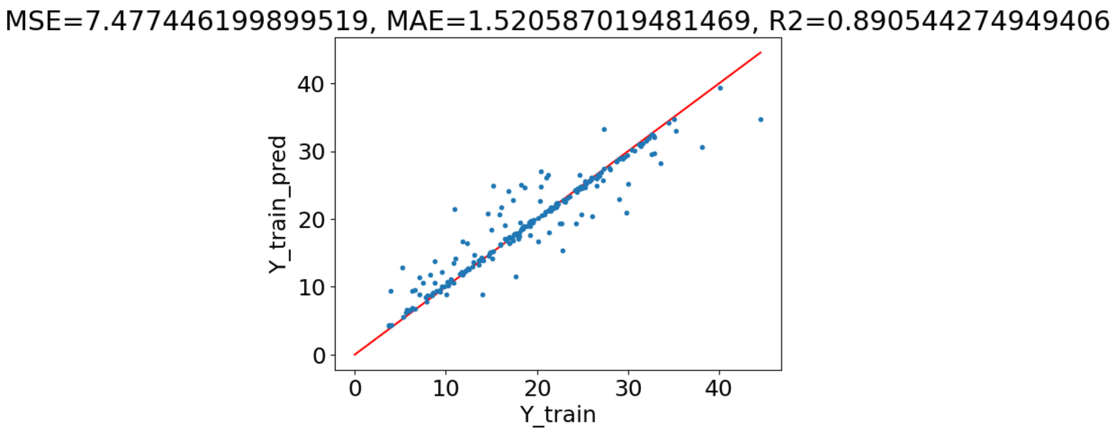
21 Training results:

[ ]:
```
from sklearn.metrics import r2_score

MSE = np.mean((Y_train - Y_train_pred)**2)
MAE = np.mean(np.abs(Y_train - Y_train_pred))
R2 = r2_score(Y_train, Y_train_pred)
#
ymax=np.max([Y_train.max(), Y_train_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_train, Y_train_pred, '.')
plt.xlabel('Y_train')
plt.ylabel('Y_train_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

```
[ ]: Text(0.5, 1.0, 'MSE=7.477446199899519, MAE=1.520587019481469,
     R2=0.890544274949406')
```

```
[WARNING] [2023-05-04 23:46:08,137:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,183:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,187:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,199:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,203:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,213:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,224:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,246:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,254:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,280:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,284:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,289:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,294:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,298:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,308:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,311:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,321:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,325:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,329:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,335:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,341:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,350:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
```

```
[WARNING] [2023-05-04 23:46:08,359:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,487:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,490:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,495:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,499:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,502:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,506:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,513:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,517:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,521:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,530:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,533:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,537:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,543:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,547:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,559:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,563:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,572:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,577:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,583:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,588:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,593:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,604:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:08,613:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
```



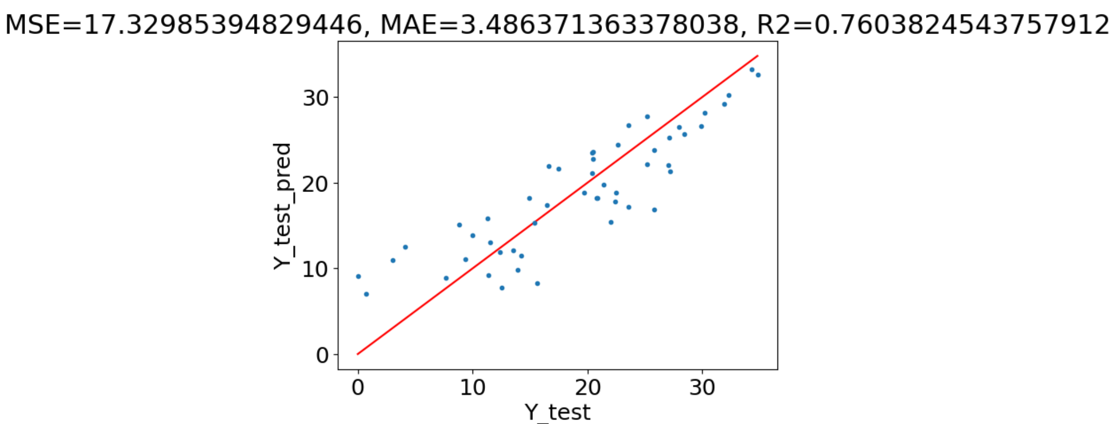
22 Testing results:

```
[ ]: MSE = np.mean((Y_test - Y_test_pred)**2)
MAE = np.mean(np.abs(Y_test - Y_test_pred))
R2 = r2_score(Y_test, Y_test_pred)
#
ymax=np.max([Y_test.max(), Y_test_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_test, Y_test_pred, '.')
plt.xlabel('Y_test')
plt.ylabel('Y_test_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

```
[ ]: Text(0.5, 1.0, 'MSE=17.32985394829446, MAE=3.486371363378038,
R2=0.7603824543757912')

[WARNING] [2023-05-04 23:46:09,349:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,352:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,358:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,362:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,377:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,386:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,393:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,396:matplotlib.font_manager] findfont: Font
```

```
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,423:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,427:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,432:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,436:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,447:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,450:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,462:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,468:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,473:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,479:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,491:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,499:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,656:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,659:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,664:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,668:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,672:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,679:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,684:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,687:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,700:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,705:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,711:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,716:matplotlib.font_manager] findfont: Font
```

```
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,726:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,732:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,742:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,747:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,751:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,758:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,772:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
[WARNING] [2023-05-04 23:46:09,779:matplotlib.font_manager] findfont: Font
family 'Times New Roman' not found.
```



23 Conclusion part 4:

This model has shown good performance overall, and it performed exceptionally well on the training set, but when we tested it on the test set, the networks we trained in part 3 outperformed it.

---

24 PyTorch Tabular

For part 5, we will use the supervised models from PyTorch Tabular.

[ ]:
```
!pip install pytorch_tabular
```

[1]:
```python
from IPython import display
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits import mplot3d
import torch
from sklearn.model_selection import train_test_split
import pandas as pd
```

[2]:
```python
X = pd.read_csv('X.csv')
Y = pd.read_csv('Y.csv')
```

[3]:
```python
data = X.join(Y)
num_col_names = data.columns.tolist()
cat_col_names = []
```

[4]:
```python
del num_col_names[-1]
```

[5]:
```python
num_col_names
```

[5]:
```
['Age',
 'Weight',
 'Height',
 'Neck',
 'Chest',
 'Abdomen',
 'Hip',
 'Thigh',
 'Knee',
 'Ankle',
 'Biceps',
 'Forearm',
 'Wrist',
 'BMI',
 'BMI/Abdomen',
 'Abdomen/Weight',
 'Chest/Abdomen',
 'Wrist/Forearm',
 'Ankle/Knee',
 'Hip/Ankle']
```

[6]:
```python
data.head()
```

[6]:
```
    Age  Weight  Height  Neck  Chest  Abdomen    Hip  Thigh  Knee  Ankle  …  \
0  23.0  154.25   67.75  36.2   93.1     85.2   94.5   59.0  37.3   21.9  …
1  22.0  173.25   72.25  38.5   93.6     83.0   98.7   58.7  37.3   23.4  …
2  22.0  154.00   66.25  34.0   95.8     87.9   99.2   59.6  38.9   24.0  …
3  26.0  184.75   72.25  37.4  101.8     86.4  101.2   60.1  37.3   22.8  …
```

```
4  24.0  184.25   71.25  34.4   97.3    100.0  101.9   63.2  42.2   24.0  …

   Forearm  Wrist       BMI  BMI/Abdomen  Abdomen/Weight  Chest/Abdomen  \
0     27.4   17.1  0.033605     0.000394        1.810446       1.092723
1     28.9   18.2  0.033189     0.000400        2.087349       1.127711
2     25.2   16.6  0.035087     0.000399        1.751991       1.089875
3     29.4   18.2  0.035392     0.000410        2.138310       1.178241
4     27.7   17.7  0.036294     0.000363        1.842500       0.973000

   Wrist/Forearm  Ankle/Knee  Hip/Ankle  BodyFat
0       0.624088    0.587131   4.315068     12.3
1       0.629758    0.627346   4.217949      6.1
2       0.658730    0.616967   4.133333     25.3
3       0.619048    0.611260   4.438596     10.4
4       0.638989    0.568720   4.245833     28.7

[5 rows x 21 columns]
```

[7]:
```
train, test = train_test_split(data, test_size=0.2, random_state=0)
train, val = train_test_split(train, test_size=0.1, random_state=0)
```

---

25 Category Embedding Model

Link

[ ]:
```
from pytorch_tabular import TabularModel
from pytorch_tabular.models.common.heads import LinearHeadConfig
from pytorch_tabular.models import CategoryEmbeddingModelConfig
from pytorch_tabular.config import (
    DataConfig,
    OptimizerConfig,
    TrainerConfig,
)

data_config = DataConfig(
    target=[
        "BodyFat"
    ],  # target should always be a list. Multi-targets are only supported for␣
 ↪regression. Multi-Task Classification is not implemented
    continuous_cols=num_col_names,
    categorical_cols=cat_col_names,
)
trainer_config = TrainerConfig(
    auto_lr_find=True, # Runs the LRFinder to automatically derive a learning␣
 ↪rate
```

```python
    batch_size=64,
    min_epochs=20,
    max_epochs=100,
    accelerator="auto", # can be 'cpu','gpu', 'tpu', or 'ipu'
)
optimizer_config = OptimizerConfig()


head_config = LinearHeadConfig(
    #layers='32', # No additional layer in head, just a mapping layer to␣
 ↪output_dim
    #activation="Softplus",
    dropout=0.0,
    initialization="kaiming",
    use_batch_norm=False
).__dict__ # Convert to dict to pass to the model config (OmegaConf doesn't␣
 ↪accept objects)

model_config = CategoryEmbeddingModelConfig(
    task="regression",
    use_batch_norm =False,
    layers="32-16-1", # Number of nodes in each layer
    activation="LeakyReLU", # Activation between each layers
    dropout=0.0,
    initialization="kaiming",
    head = "LinearHead", #Linear Head
    head_config = head_config, # Linear Head Config
    learning_rate = 1e-3
)


tabular_model = TabularModel(
    data_config=data_config,
    model_config=model_config,
    optimizer_config=optimizer_config,
    trainer_config=trainer_config,
)

tabular_model.fit(train=train, validation=val)
result = tabular_model.evaluate(test)
pred_df = tabular_model.predict(test)
```

```
2023-05-04 23:54:47,360 - {pytorch_tabular.tabular_model:102} - INFO -
Experiment Tracking is turned off
INFO:pytorch_tabular.tabular_model:Experiment Tracking is turned off
INFO:lightning_lite.utilities.seed:Global seed set to 42
2023-05-04 23:54:47,391 - {pytorch_tabular.tabular_model:465} - INFO - Preparing
```

```
the DataLoaders
INFO:pytorch_tabular.tabular_model:Preparing the DataLoaders
2023-05-04 23:54:47,396 - {pytorch_tabular.tabular_datamodule:286} - INFO -
Setting up the datamodule for regression task
INFO:pytorch_tabular.tabular_datamodule:Setting up the datamodule for regression
task
2023-05-04 23:54:47,417 - {pytorch_tabular.tabular_model:508} - INFO - Preparing
the Model: CategoryEmbeddingModel
INFO:pytorch_tabular.tabular_model:Preparing the Model: CategoryEmbeddingModel
2023-05-04 23:54:47,450 - {pytorch_tabular.tabular_model:264} - INFO - Preparing
the Trainer
INFO:pytorch_tabular.tabular_model:Preparing the Trainer
INFO:pytorch_lightning.utilities.rank_zero:GPU available: False, used: False
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TPU
cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HPUs
2023-05-04 23:54:47,499 - {pytorch_tabular.tabular_model:558} - INFO - Auto LR
Find Started
INFO:pytorch_tabular.tabular_model:Auto LR Find Started
/usr/local/lib/python3.10/dist-
packages/pytorch_lightning/callbacks/model_checkpoint.py:604: UserWarning:
Checkpoint directory /content/saved_models exists and is not empty.
  rank_zero_warn(f"Checkpoint directory {dirpath} exists and is not empty.")

Finding best initial lr:   0%|          | 0/100 [00:00<?, ?it/s]

INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped:
`max_steps=100` reached.
INFO:pytorch_lightning.tuner.lr_finder:Learning rate set to 0.2754228703338169
INFO:pytorch_lightning.utilities.rank_zero:Restoring states from the checkpoint
path at /content/.lr_find_5a1b230c-480f-4f6d-9a3f-acab0954f5ad.ckpt
INFO:pytorch_lightning.utilities.rank_zero:Restored all states from the
checkpoint file at /content/.lr_find_5a1b230c-480f-4f6d-9a3f-acab0954f5ad.ckpt
2023-05-04 23:54:48,750 - {pytorch_tabular.tabular_model:560} - INFO - Suggested
LR: 0.2754228703338169. For plot and detailed analysis, use `find_learning_rate`
method.
INFO:pytorch_tabular.tabular_model:Suggested LR: 0.2754228703338169. For plot
and detailed analysis, use `find_learning_rate` method.
2023-05-04 23:54:48,757 - {pytorch_tabular.tabular_model:566} - INFO - Training
Started
INFO:pytorch_tabular.tabular_model:Training Started
```

	Name	Type	Params
0	_backbone	CategoryEmbeddingBackbone	1.2 K
1	_embedding_layer	Embedding1dLayer	40
2	head	LinearHead	2

```
  3   loss               MSELoss                          0
```

```
Trainable params: 1.3 K
Non-trainable params: 0
Total params: 1.3 K
Total estimated model params size (MB): 0
```

```
Output()
```

```
INFO:pytorch_lightning.utilities.rank_zero:Trainer was signaled to stop but the
required `min_epochs=20` or `min_steps=None` has not been met. Training will
continue…
```

```
2023-05-04 23:54:53,113 - {pytorch_tabular.tabular_model:568} - INFO - Training
the model completed
INFO:pytorch_tabular.tabular_model:Training the model completed
2023-05-04 23:54:53,117 - {pytorch_tabular.tabular_model:1207} - INFO - Loading
the best model
INFO:pytorch_tabular.tabular_model:Loading the best model
```

```
Output()
```

Test metric	DataLoader 0
test_loss	17.65951919555664
test_mean_squared_error	17.65951919555664

```
Output()
```

[ ]:
```
pred_df.head()
```

[ ]:
```
      Age  Weight  Height  Neck  Chest  Abdomen   Hip  Thigh  Knee  Ankle  \
158  30.0  136.50   68.75  35.9   88.7     76.6  89.8   50.1  34.8   21.8
```

```
83   70.0  170.75   70.00  38.7  101.8     94.9  95.0   56.0  36.5   24.1
170  35.0  152.25   67.75  37.0   92.2     81.9  92.8   54.7  36.2   22.1
101  48.0  173.75   72.00  37.0   99.1     92.0  98.3   59.3  38.4   22.4
150  26.0  152.25   69.00  35.4   92.9     77.6  93.5   56.9  35.9   20.4

     …  Wrist       BMI  BMI/Abdomen  Abdomen/Weight  Chest/Abdomen  \
158  …   16.9  0.028879     0.000377        1.781984       1.157963
83   …   19.2  0.034847     0.000367        1.799262       1.072708
170  …   17.7  0.033169     0.000405        1.858974       1.125763
101  …   17.0  0.033517     0.000364        1.888587       1.077174
150  …   17.8  0.031979     0.000412        1.961985       1.197165

     Wrist/Forearm  Ankle/Knee  Hip/Ankle  BodyFat  BodyFat_prediction
158       0.496329    0.626437   4.119266     12.5            6.937536
83        0.703297    0.660274   3.941909     27.0           19.689533
170       0.645985    0.610497   4.199095      3.0           10.559484
101       0.648855    0.583333   4.388393     20.4           19.139286
150       0.613793    0.568245   4.583333      9.4            6.316311

[5 rows x 22 columns]
```

[ ]:
```python
Y_train = train['BodyFat'].values
Y_test = test['BodyFat'].values

Y_train_pred = tabular_model.predict(train)["BodyFat_prediction"].values
Y_test_pred = tabular_model.predict(test)["BodyFat_prediction"].values
```

```
Output()
```

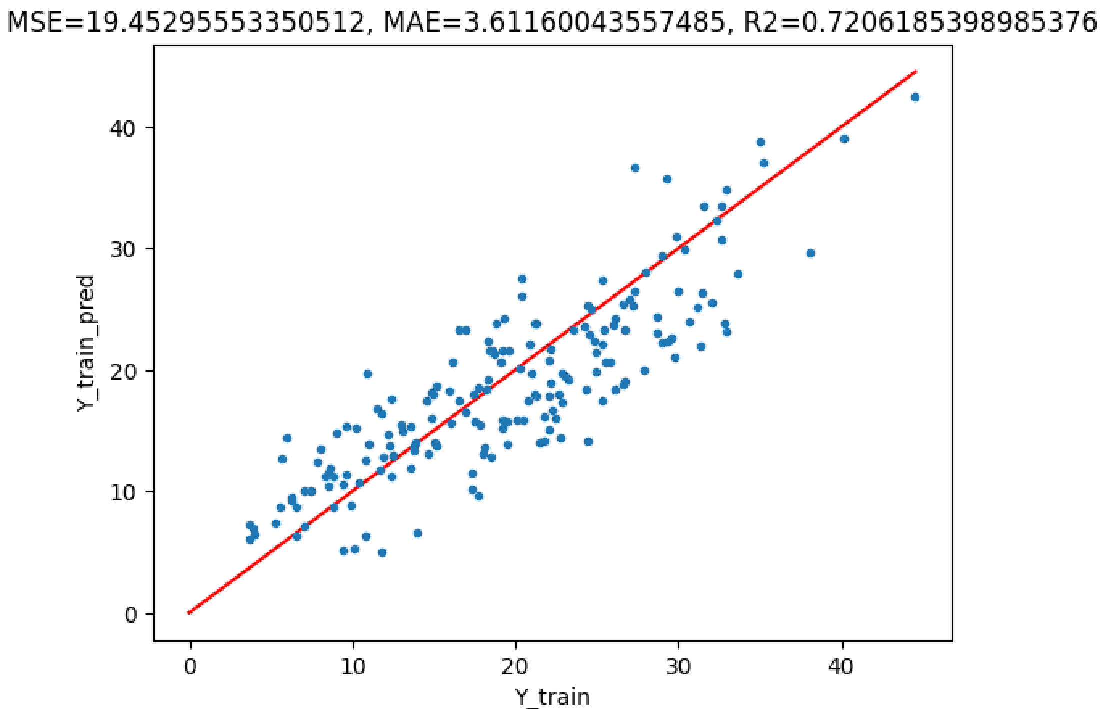
```
Output()
```

26 Training results:

[ ]:
```python
from sklearn.metrics import r2_score

MSE = np.mean((Y_train - Y_train_pred)**2)
MAE = np.mean(np.abs(Y_train - Y_train_pred))
R2 = r2_score(Y_train, Y_train_pred)
#
```

```
ymax=np.max([Y_train.max(), Y_train_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_train, Y_train_pred, '.')
plt.xlabel('Y_train')
plt.ylabel('Y_train_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

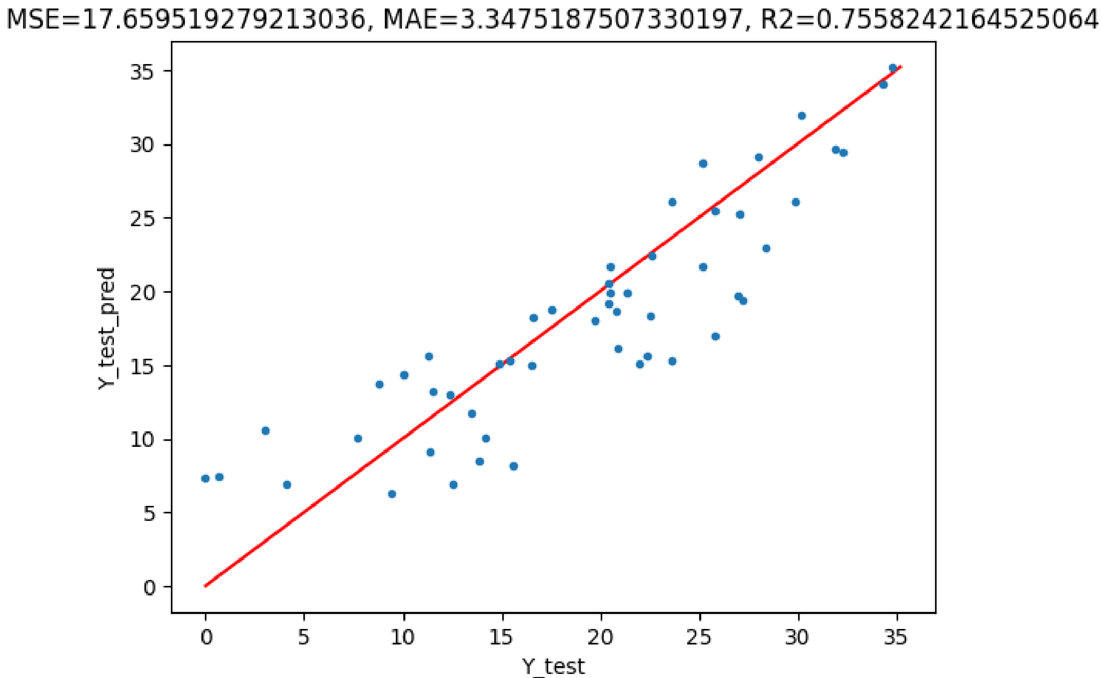
[ ]: Text(0.5, 1.0, 'MSE=19.45295553350512, MAE=3.61160043557485, R2=0.7206185398985376')



27 Testing results:

[ ]:
```
MSE = np.mean((Y_test - Y_test_pred)**2)
MAE = np.mean(np.abs(Y_test - Y_test_pred))
R2 = r2_score(Y_test, Y_test_pred)
#
ymax=np.max([Y_test.max(), Y_test_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_test, Y_test_pred, '.')
plt.xlabel('Y_test')
plt.ylabel('Y_test_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[ ]: Text(0.5, 1.0, 'MSE=17.659519279213036, MAE=3.3475187507330197,
R2=0.7558242164525064')



---

28 Gated Additive Tree Ensemble (GATE)

Link

[17]:
```
from pytorch_tabular import TabularModel
from pytorch_tabular.models.common.heads import LinearHeadConfig
from pytorch_tabular.models import GatedAdditiveTreeEnsembleConfig
from pytorch_tabular.config import (
    DataConfig,
    OptimizerConfig,
    TrainerConfig,
)

data_config = DataConfig(
    target=[
        "BodyFat"
    ],  # target should always be a list. Multi-targets are only supported for␣
 ↪regression. Multi-Task Classification is not implemented
    continuous_cols=num_col_names,
    categorical_cols=cat_col_names,
```

```
)
trainer_config = TrainerConfig(
    auto_lr_find=True, # Runs the LRFinder to automatically derive a learning␣
 ↪rate
    batch_size=64,
    min_epochs=100,
    max_epochs=200,
    accelerator="auto", # can be 'cpu','gpu', 'tpu', or 'ipu'
)
optimizer_config = OptimizerConfig()


head_config = LinearHeadConfig(
    #layers='32', # No additional layer in head, just a mapping layer to␣
 ↪output_dim
    #activation="Softplus",
    dropout=0.0,
    initialization="kaiming",
    use_batch_norm=False
).__dict__ # Convert to dict to pass to the model config (OmegaConf doesn't␣
 ↪accept objects)

model_config = GatedAdditiveTreeEnsembleConfig(
    task="regression",
    head = "LinearHead", #Linear Head
    head_config = head_config, # Linear Head Config
    learning_rate = 1e-3
)


tabular_model = TabularModel(
    data_config=data_config,
    model_config=model_config,
    optimizer_config=optimizer_config,
    trainer_config=trainer_config,
)

tabular_model.fit(train=train, validation=val)
result = tabular_model.evaluate(test)
pred_df = tabular_model.predict(test)
```

```
2023-05-05 19:23:38,807 - {pytorch_tabular.tabular_model:102} - INFO -
Experiment Tracking is turned off
INFO:pytorch_tabular.tabular_model:Experiment Tracking is turned off
INFO:lightning_lite.utilities.seed:Global seed set to 42
2023-05-05 19:23:38,845 - {pytorch_tabular.tabular_model:465} - INFO - Preparing
the DataLoaders
```

```
INFO:pytorch_tabular.tabular_model:Preparing the DataLoaders
2023-05-05 19:23:38,857 - {pytorch_tabular.tabular_datamodule:286} - INFO -
Setting up the datamodule for regression task
INFO:pytorch_tabular.tabular_datamodule:Setting up the datamodule for regression
task
2023-05-05 19:23:38,899 - {pytorch_tabular.tabular_model:508} - INFO - Preparing
the Model: GatedAdditiveTreeEnsembleModel
INFO:pytorch_tabular.tabular_model:Preparing the Model:
GatedAdditiveTreeEnsembleModel
/usr/local/lib/python3.10/dist-
packages/pytorch_tabular/models/base_model.py:126: UserWarning: Wandb is not
installed. Please install wandb to log logits. You can install wandb using pip
install wandb or install PyTorch Tabular using pip install pytorch-tabular[all]
  warnings.warn(
2023-05-05 19:23:39,480 - {pytorch_tabular.models.gate.gate_model:221} - INFO -
Data Aware Initialization of T0
INFO:pytorch_tabular.models.gate.gate_model:Data Aware Initialization of T0
2023-05-05 19:23:39,506 - {pytorch_tabular.tabular_model:264} - INFO - Preparing
the Trainer
INFO:pytorch_tabular.tabular_model:Preparing the Trainer
INFO:pytorch_lightning.utilities.rank_zero:GPU available: False, used: False
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TPU
cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HPUs
2023-05-05 19:23:39,591 - {pytorch_tabular.tabular_model:558} - INFO - Auto LR
Find Started
INFO:pytorch_tabular.tabular_model:Auto LR Find Started
/usr/local/lib/python3.10/dist-
packages/pytorch_lightning/callbacks/model_checkpoint.py:604: UserWarning:
Checkpoint directory /content/saved_models exists and is not empty.
  rank_zero_warn(f"Checkpoint directory {dirpath} exists and is not empty.")

Finding best initial lr:   0%|          | 0/100 [00:00<?, ?it/s]

INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped:
`max_steps=100` reached.
INFO:pytorch_lightning.tuner.lr_finder:Learning rate set to 0.19054607179632482
INFO:pytorch_lightning.utilities.rank_zero:Restoring states from the checkpoint
path at /content/.lr_find_4553d0c8-c32b-4b06-8e5f-07e9c325277f.ckpt
INFO:pytorch_lightning.utilities.rank_zero:Restored all states from the
checkpoint file at /content/.lr_find_4553d0c8-c32b-4b06-8e5f-07e9c325277f.ckpt
2023-05-05 19:28:24,874 - {pytorch_tabular.tabular_model:560} - INFO - Suggested
LR: 0.19054607179632482. For plot and detailed analysis, use
`find_learning_rate` method.
INFO:pytorch_tabular.tabular_model:Suggested LR: 0.19054607179632482. For plot
and detailed analysis, use `find_learning_rate` method.
2023-05-05 19:28:24,879 - {pytorch_tabular.models.gate.gate_model:221} - INFO -
Data Aware Initialization of T0
```

```
INFO:pytorch_tabular.models.gate.gate_model:Data Aware Initialization of T0
2023-05-05 19:28:24,899 - {pytorch_tabular.tabular_model:566} - INFO - Training
Started
INFO:pytorch_tabular.tabular_model:Training Started
/usr/local/lib/python3.10/dist-
packages/pytorch_lightning/callbacks/model_checkpoint.py:604: UserWarning:
Checkpoint directory /content/saved_models exists and is not empty.
  rank_zero_warn(f"Checkpoint directory {dirpath} exists and is not empty.")
```

	Name	Type	Params
0	_backbone	GatedAdditiveTreesBackbone	1.1 M
1	_embedding_layer	Embedding1dLayer	40
2	_head	CustomHead	54
3	loss	MSELoss	0

Trainable params: 1.1 M
Non-trainable params: 0
Total params: 1.1 M
Total estimated model params size (MB): 4

```
Output()
```

```
INFO:pytorch_lightning.utilities.rank_zero:Trainer was signaled to stop but the
required `min_epochs=100` or `min_steps=None` has not been met. Training will
continue…
```

```
2023-05-05 19:45:17,462 - {pytorch_tabular.tabular_model:568} - INFO - Training
the model completed
INFO:pytorch_tabular.tabular_model:Training the model completed
2023-05-05 19:45:17,471 - {pytorch_tabular.tabular_model:1207} - INFO - Loading
the best model
INFO:pytorch_tabular.tabular_model:Loading the best model
```

```
Output()
```

Test metric	DataLoader 0
test_loss	25.315202713012695
test_mean_squared_error	25.315202713012695

```
Output()
```

```
[21]: Y_train = train['BodyFat'].values
Y_test = test['BodyFat'].values

Y_train_pred = tabular_model.predict(train)["BodyFat_prediction"].values
Y_test_pred = tabular_model.predict(test)["BodyFat_prediction"].values
```

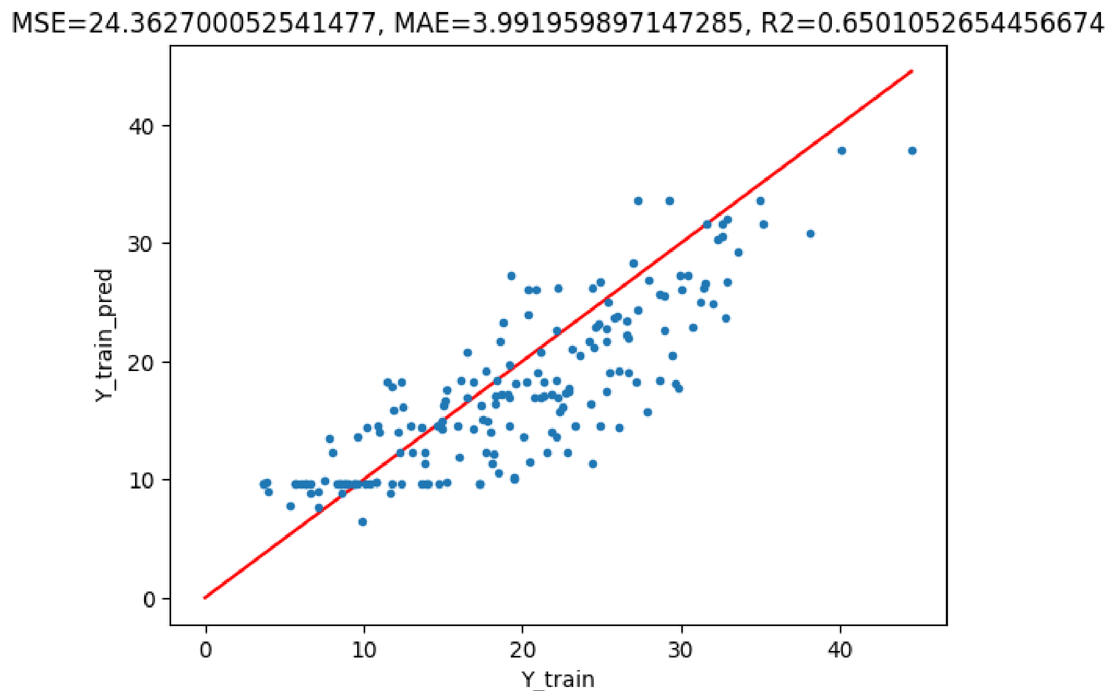
```
Output()
```

```
Output()
```

29 Training results:

```
[22]: from sklearn.metrics import r2_score

MSE = np.mean((Y_train - Y_train_pred)**2)
MAE = np.mean(np.abs(Y_train - Y_train_pred))
R2 = r2_score(Y_train, Y_train_pred)
#
ymax=np.max([Y_train.max(), Y_train_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_train, Y_train_pred, '.')
plt.xlabel('Y_train')
plt.ylabel('Y_train_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

```
[22]: Text(0.5, 1.0, 'MSE=24.362700052541477, MAE=3.991959897147285,
R2=0.6501052654456674')
```

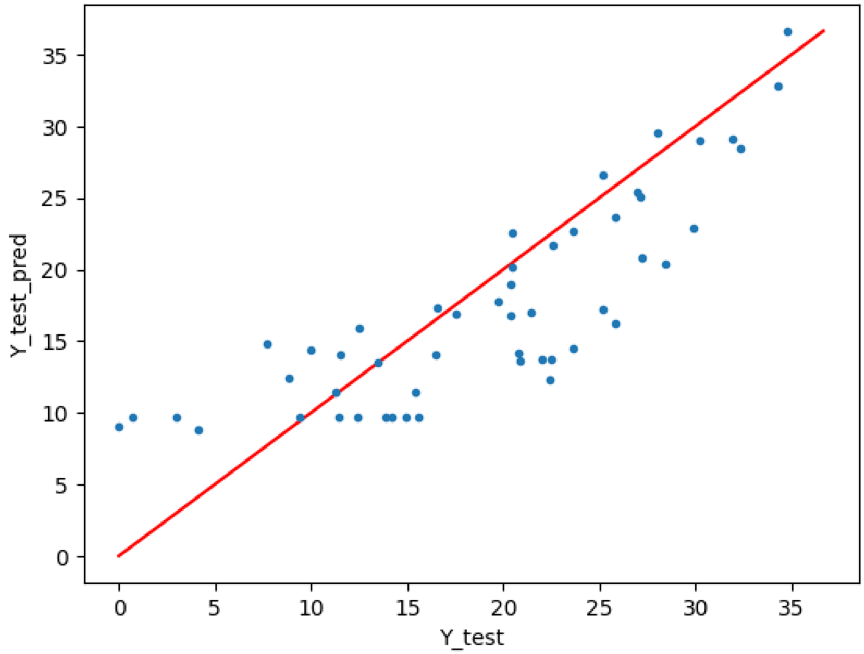


30 Testing results:

[23]:
```
MSE = np.mean((Y_test - Y_test_pred)**2)
MAE = np.mean(np.abs(Y_test - Y_test_pred))
R2 = r2_score(Y_test, Y_test_pred)
#
ymax=np.max([Y_test.max(), Y_test_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_test, Y_test_pred, '.')
plt.xlabel('Y_test')
plt.ylabel('Y_test_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[23]:
```
Text(0.5, 1.0, 'MSE=25.315202180551083, MAE=4.075024017633176,
R2=0.6499701248733687')
```

MSE=25.315202180551083, MAE=4.075024017633176, R2=0.6499701248733687



---

31 Neural Oblivious Decision Ensembles (NODE)

[12]:
```
from pytorch_tabular import TabularModel
from pytorch_tabular.models.common.heads import LinearHeadConfig
from pytorch_tabular.models import NodeConfig
from pytorch_tabular.config import (
    DataConfig,
    OptimizerConfig,
    TrainerConfig,
)

data_config = DataConfig(
    target=[
        "BodyFat"
    ],  # target should always be a list. Multi-targets are only supported for␣
↪regression. Multi-Task Classification is not implemented
    continuous_cols=num_col_names,
    categorical_cols=cat_col_names,
)
trainer_config = TrainerConfig(
    auto_lr_find=True, # Runs the LRFinder to automatically derive a learning␣
↪rate
```

```
    batch_size=64,
    min_epochs=200,
    max_epochs=1000,
    accelerator="auto", # can be 'cpu','gpu', 'tpu', or 'ipu'
)
optimizer_config = OptimizerConfig()


head_config = LinearHeadConfig(
    layers='32', # No additional layer in head, just a mapping layer to␣
 ↪output_dim
    activation="Softplus",
    dropout=0.0,
    initialization="kaiming",
    use_batch_norm=False
).__dict__ # Convert to dict to pass to the model config (OmegaConf doesn't␣
 ↪accept objects)

model_config = NodeConfig(
    task="regression",
    head = "LinearHead", #Linear Head
    head_config = head_config, # Linear Head Config
    learning_rate = 1e-3,
    num_layers = 3,
    num_trees = 64,
    depth = 3,
    batch_norm_continuous_input = True

)


tabular_model = TabularModel(
    data_config=data_config,
    model_config=model_config,
    optimizer_config=optimizer_config,
    trainer_config=trainer_config,
)

tabular_model.fit(train=train, validation=val)
```

```
/usr/local/lib/python3.10/dist-
packages/pytorch_tabular/models/node/config.py:218: UserWarning:
embed_categorical is set to False and will use LeaveOneOutEncoder to encode
categorical features. This is deprecated and will be removed in future versions
and categorical columns will be embedded by default.
  warnings.warn(
2023-05-05 19:18:54,013 - {pytorch_tabular.tabular_model:102} - INFO -
```

```
Experiment Tracking is turned off
INFO:pytorch_tabular.tabular_model:Experiment Tracking is turned off
INFO:lightning_lite.utilities.seed:Global seed set to 42
2023-05-05 19:18:54,059 - {pytorch_tabular.tabular_model:465} - INFO - Preparing
the DataLoaders
INFO:pytorch_tabular.tabular_model:Preparing the DataLoaders
2023-05-05 19:18:54,064 - {pytorch_tabular.tabular_datamodule:286} - INFO -
Setting up the datamodule for regression task
INFO:pytorch_tabular.tabular_datamodule:Setting up the datamodule for regression
task
2023-05-05 19:18:54,094 - {pytorch_tabular.tabular_model:508} - INFO - Preparing
the Model: NODEModel
INFO:pytorch_tabular.tabular_model:Preparing the Model: NODEModel
/usr/local/lib/python3.10/dist-
packages/pytorch_tabular/models/node/node_model.py:116: UserWarning: Ignoring
head config because NODE has a specific head which subsets the tree outputs
  warnings.warn("Ignoring head config because NODE has a specific head which
subsets the tree outputs")
/usr/local/lib/python3.10/dist-
packages/pytorch_tabular/models/base_model.py:126: UserWarning: Wandb is not
installed. Please install wandb to log logits. You can install wandb using pip
install wandb or install PyTorch Tabular using pip install pytorch-tabular[all]
  warnings.warn(
2023-05-05 19:18:54,141 - {pytorch_tabular.models.node.node_model:82} - INFO -
Data Aware Initialization of NODE using a forward pass with 2000 batch size…
INFO:pytorch_tabular.models.node.node_model:Data Aware Initialization of NODE
using a forward pass with 2000 batch size…
/usr/local/lib/python3.10/dist-packages/pytorch_tabular/models/node/odst.py:143:
UserWarning: Data-aware initialization is performed on less than 1000 data
points. This may cause instability.To avoid potential problems, run this model
on a data batch with at least 1000 data samples.You can do so manually before
training. Use with torch.no_grad() for memory efficiency.
  warn(
2023-05-05 19:18:54,342 - {pytorch_tabular.tabular_model:264} - INFO - Preparing
the Trainer
INFO:pytorch_tabular.tabular_model:Preparing the Trainer
INFO:pytorch_lightning.utilities.rank_zero:GPU available: False, used: False
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TPU
cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HPUs
2023-05-05 19:18:54,421 - {pytorch_tabular.tabular_model:558} - INFO - Auto LR
Find Started
INFO:pytorch_tabular.tabular_model:Auto LR Find Started
/usr/local/lib/python3.10/dist-
packages/pytorch_lightning/callbacks/model_checkpoint.py:604: UserWarning:
Checkpoint directory /content/saved_models exists and is not empty.
  rank_zero_warn(f"Checkpoint directory {dirpath} exists and is not empty.")
```

```
Finding best initial lr:   0%|          | 0/100 [00:00<?, ?it/s]

INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped:
`max_steps=100` reached.
INFO:pytorch_lightning.tuner.lr_finder:Learning rate set to 0.8317637711026709
INFO:pytorch_lightning.utilities.rank_zero:Restoring states from the checkpoint
path at /content/.lr_find_43e70b70-76a5-45d4-9a2a-6b13615dbf75.ckpt
INFO:pytorch_lightning.utilities.rank_zero:Restored all states from the
checkpoint file at /content/.lr_find_43e70b70-76a5-45d4-9a2a-6b13615dbf75.ckpt
2023-05-05 19:19:01,531 - {pytorch_tabular.tabular_model:560} - INFO - Suggested
LR: 0.8317637711026709. For plot and detailed analysis, use `find_learning_rate`
method.
INFO:pytorch_tabular.tabular_model:Suggested LR: 0.8317637711026709. For plot
and detailed analysis, use `find_learning_rate` method.
2023-05-05 19:19:01,537 - {pytorch_tabular.models.node.node_model:82} - INFO -
Data Aware Initialization of NODE using a forward pass with 2000 batch size…
INFO:pytorch_tabular.models.node.node_model:Data Aware Initialization of NODE
using a forward pass with 2000 batch size…
2023-05-05 19:19:01,579 - {pytorch_tabular.tabular_model:566} - INFO - Training
Started
INFO:pytorch_tabular.tabular_model:Training Started
/usr/local/lib/python3.10/dist-
packages/pytorch_lightning/callbacks/model_checkpoint.py:604: UserWarning:
Checkpoint directory /content/saved_models exists and is not empty.
  rank_zero_warn(f"Checkpoint directory {dirpath} exists and is not empty.")
```

	Name	Type	Params
0	_backbone	NODEBackbone	166 K
1	_embedding_layer	PreEncoded1dLayer	40
2	_head	Lambda	0
3	loss	MSELoss	0

Trainable params: 166 K
Non-trainable params: 147
Total params: 166 K
Total estimated model params size (MB): 0

```
Output()

INFO:pytorch_lightning.utilities.rank_zero:Trainer was signaled to stop but the
required `min_epochs=200` or `min_steps=None` has not been met. Training will
continue…
```

```
2023-05-05 19:20:54,135 - {pytorch_tabular.tabular_model:568} - INFO - Training
the model completed
INFO:pytorch_tabular.tabular_model:Training the model completed
2023-05-05 19:20:54,139 - {pytorch_tabular.tabular_model:1207} - INFO - Loading
the best model
INFO:pytorch_tabular.tabular_model:Loading the best model
```

[12]: <pytorch_lightning.trainer.trainer.Trainer at 0x7fca38d15870>

[14]:
```
Y_train = train['BodyFat'].values
Y_test = test['BodyFat'].values

Y_train_pred = tabular_model.predict(train)["BodyFat_prediction"].values
Y_test_pred = tabular_model.predict(test)["BodyFat_prediction"].values
```

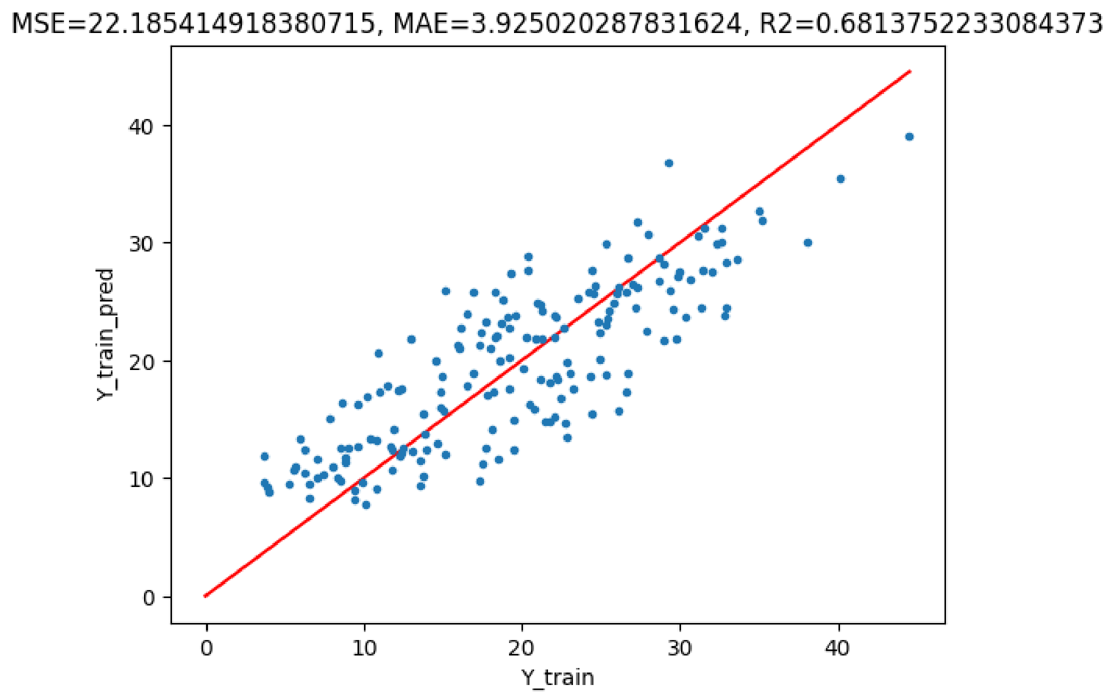
```
Output()
```

```
Output()
```

32 Training results:

[15]:
```
from sklearn.metrics import r2_score

MSE = np.mean((Y_train - Y_train_pred)**2)
MAE = np.mean(np.abs(Y_train - Y_train_pred))
R2 = r2_score(Y_train, Y_train_pred)
#
ymax=np.max([Y_train.max(), Y_train_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_train, Y_train_pred, '.')
plt.xlabel('Y_train')
plt.ylabel('Y_train_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

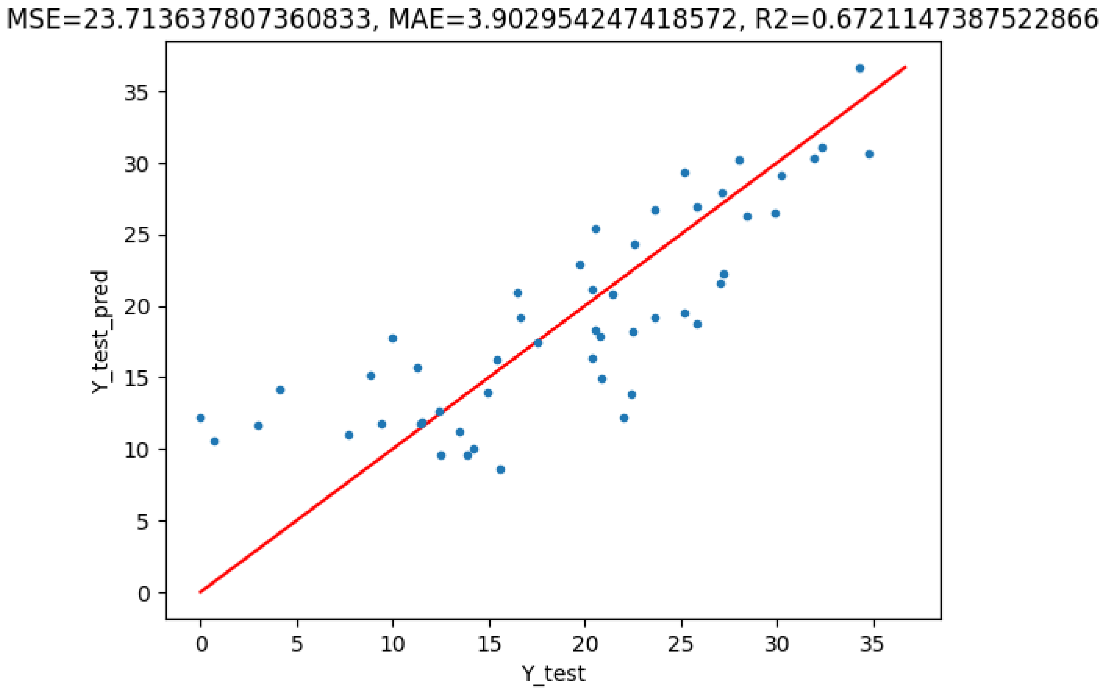
[15]:
```
Text(0.5, 1.0, 'MSE=22.185414918380715, MAE=3.925020287831624,
R2=0.6813752233084373')
```



33 Testing results:

```
[16]: MSE = np.mean((Y_test - Y_test_pred)**2)
      MAE = np.mean(np.abs(Y_test - Y_test_pred))
      R2 = r2_score(Y_test, Y_test_pred)
      #
      ymax=np.max([Y_test.max(), Y_test_pred.max()])
      plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
      plt.plot(Y_test, Y_test_pred, '.')
      plt.xlabel('Y_test')
      plt.ylabel('Y_test_pred')
      plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

```
[16]: Text(0.5, 1.0, 'MSE=23.713637807360833, MAE=3.902954247418572,
      R2=0.6721147387522866')
```



---

34 TabNet

Link

[39]:
```
from pytorch_tabular import TabularModel
from pytorch_tabular.models.common.heads import LinearHeadConfig
from pytorch_tabular.models import TabNetModelConfig
from pytorch_tabular.config import (
    DataConfig,
    OptimizerConfig,
    TrainerConfig,
)

data_config = DataConfig(
    target=[
        "BodyFat"
    ],  # target should always be a list. Multi-targets are only supported for␣
↪regression. Multi-Task Classification is not implemented
    continuous_cols=num_col_names,
    categorical_cols=cat_col_names,
)
trainer_config = TrainerConfig(
```

```
    auto_lr_find=True, # Runs the LRFinder to automatically derive a learning␣
↪rate
    batch_size=64,
    min_epochs=40,
    max_epochs=100,
    accelerator="auto", # can be 'cpu','gpu', 'tpu', or 'ipu'
)
optimizer_config = OptimizerConfig()


head_config = LinearHeadConfig(
    layers='32', # No additional layer in head, just a mapping layer to␣
↪output_dim
    activation="Softplus",
    dropout=0.0,
    initialization="kaiming",
    use_batch_norm=False
).__dict__ # Convert to dict to pass to the model config (OmegaConf doesn't␣
↪accept objects)

model_config = TabNetModelConfig(
    task="regression",
    head = "LinearHead", #Linear Head
    head_config = head_config, # Linear Head Config
    learning_rate = 1e-3,
    n_steps = 6,
    virtual_batch_size = 32,
    batch_norm_continuous_input = True
)


tabular_model = TabularModel(
    data_config=data_config,
    model_config=model_config,
    optimizer_config=optimizer_config,
    trainer_config=trainer_config,
)

tabular_model.fit(train=train, validation=val)
```

```
2023-05-05 19:09:08,050 - {pytorch_tabular.tabular_model:102} - INFO -
Experiment Tracking is turned off
INFO:pytorch_tabular.tabular_model:Experiment Tracking is turned off
INFO:lightning_lite.utilities.seed:Global seed set to 42
2023-05-05 19:09:08,141 - {pytorch_tabular.tabular_model:465} - INFO - Preparing
the DataLoaders
INFO:pytorch_tabular.tabular_model:Preparing the DataLoaders
```

```
2023-05-05 19:09:08,153 - {pytorch_tabular.tabular_datamodule:286} - INFO -
Setting up the datamodule for regression task
INFO:pytorch_tabular.tabular_datamodule:Setting up the datamodule for regression
task
2023-05-05 19:09:08,212 - {pytorch_tabular.tabular_model:508} - INFO - Preparing
the Model: TabNetModel
INFO:pytorch_tabular.tabular_model:Preparing the Model: TabNetModel
/usr/local/lib/python3.10/dist-
packages/pytorch_tabular/models/base_model.py:126: UserWarning: Wandb is not
installed. Please install wandb to log logits. You can install wandb using pip
install wandb or install PyTorch Tabular using pip install pytorch-tabular[all]
  warnings.warn(
2023-05-05 19:09:08,328 - {pytorch_tabular.tabular_model:264} - INFO - Preparing
the Trainer
INFO:pytorch_tabular.tabular_model:Preparing the Trainer
INFO:pytorch_lightning.utilities.rank_zero:GPU available: False, used: False
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TPU
cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HPUs
2023-05-05 19:09:08,419 - {pytorch_tabular.tabular_model:558} - INFO - Auto LR
Find Started
INFO:pytorch_tabular.tabular_model:Auto LR Find Started
/usr/local/lib/python3.10/dist-
packages/pytorch_lightning/callbacks/model_checkpoint.py:604: UserWarning:
Checkpoint directory /content/saved_models exists and is not empty.
  rank_zero_warn(f"Checkpoint directory {dirpath} exists and is not empty.")

Finding best initial lr:   0%|          | 0/100 [00:00<?, ?it/s]

INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped:
`max_steps=100` reached.
INFO:pytorch_lightning.tuner.lr_finder:Learning rate set to 0.07585775750291836
INFO:pytorch_lightning.utilities.rank_zero:Restoring states from the checkpoint
path at /content/.lr_find_fa29f07a-6432-434c-b762-ecaa7f790dd0.ckpt
INFO:pytorch_lightning.utilities.rank_zero:Restored all states from the
checkpoint file at /content/.lr_find_fa29f07a-6432-434c-b762-ecaa7f790dd0.ckpt
2023-05-05 19:09:13,831 - {pytorch_tabular.tabular_model:560} - INFO - Suggested
LR: 0.07585775750291836. For plot and detailed analysis, use
`find_learning_rate` method.
INFO:pytorch_tabular.tabular_model:Suggested LR: 0.07585775750291836. For plot
and detailed analysis, use `find_learning_rate` method.
2023-05-05 19:09:13,838 - {pytorch_tabular.tabular_model:566} - INFO - Training
Started
INFO:pytorch_tabular.tabular_model:Training Started
```

Name	Type	Params

0	_embedding_layer	Identity	0
1	_backbone	TabNetBackbone	11.4 K
2	_head	Identity	0
3	loss	MSELoss	0

Trainable params: 11.4 K
Non-trainable params: 0
Total params: 11.4 K
Total estimated model params size (MB): 0

```
Output()
```

```
INFO:pytorch_lightning.utilities.rank_zero:Trainer was signaled to stop but the
required `min_epochs=40` or `min_steps=None` has not been met. Training will
continue…
```

```
2023-05-05 19:09:33,646 - {pytorch_tabular.tabular_model:568} - INFO - Training
the model completed
INFO:pytorch_tabular.tabular_model:Training the model completed
2023-05-05 19:09:33,650 - {pytorch_tabular.tabular_model:1207} - INFO - Loading
the best model
INFO:pytorch_tabular.tabular_model:Loading the best model
```

[39]:
```
<pytorch_lightning.trainer.trainer.Trainer at 0x7fc6b9fbc7f0>
```

[40]:
```python
Y_train = train['BodyFat'].values
Y_test = test['BodyFat'].values

Y_train_pred = tabular_model.predict(train)["BodyFat_prediction"].values
Y_test_pred = tabular_model.predict(test)["BodyFat_prediction"].values
```

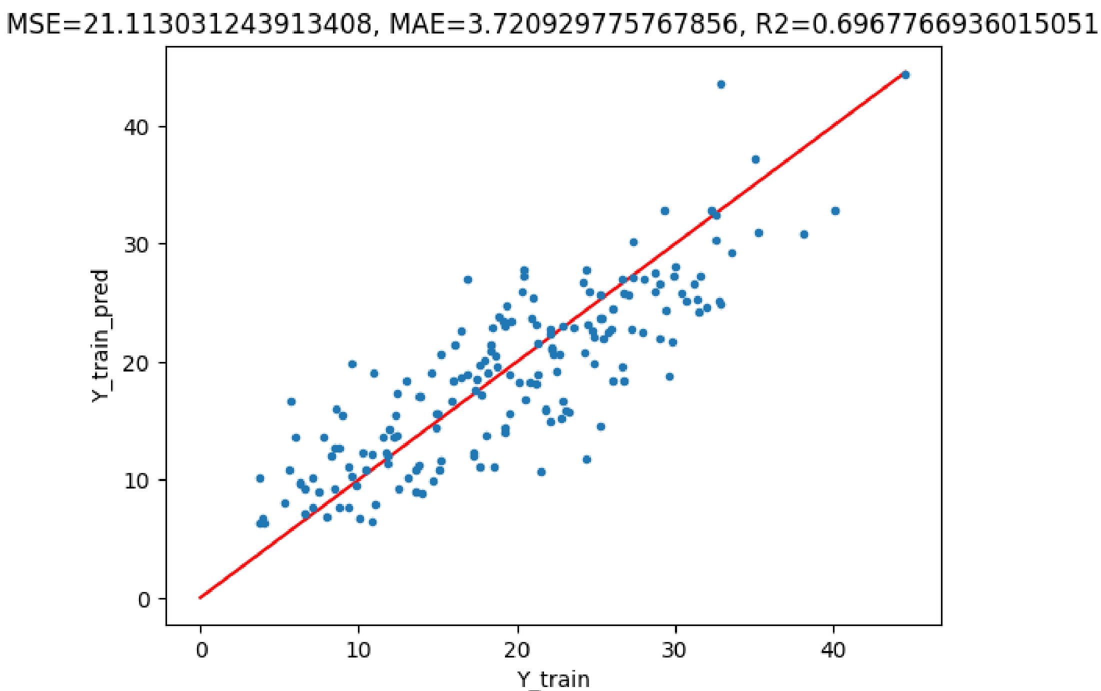
```
Output()
```

```
Output()
```

35 Training results:

[41]:
```
from sklearn.metrics import r2_score

MSE = np.mean((Y_train - Y_train_pred)**2)
MAE = np.mean(np.abs(Y_train - Y_train_pred))
R2 = r2_score(Y_train, Y_train_pred)
#
ymax=np.max([Y_train.max(), Y_train_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_train, Y_train_pred, '.')
plt.xlabel('Y_train')
plt.ylabel('Y_train_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[41]:
```
Text(0.5, 1.0, 'MSE=21.113031243913408, MAE=3.720929775767856,
R2=0.6967766936015051')
```

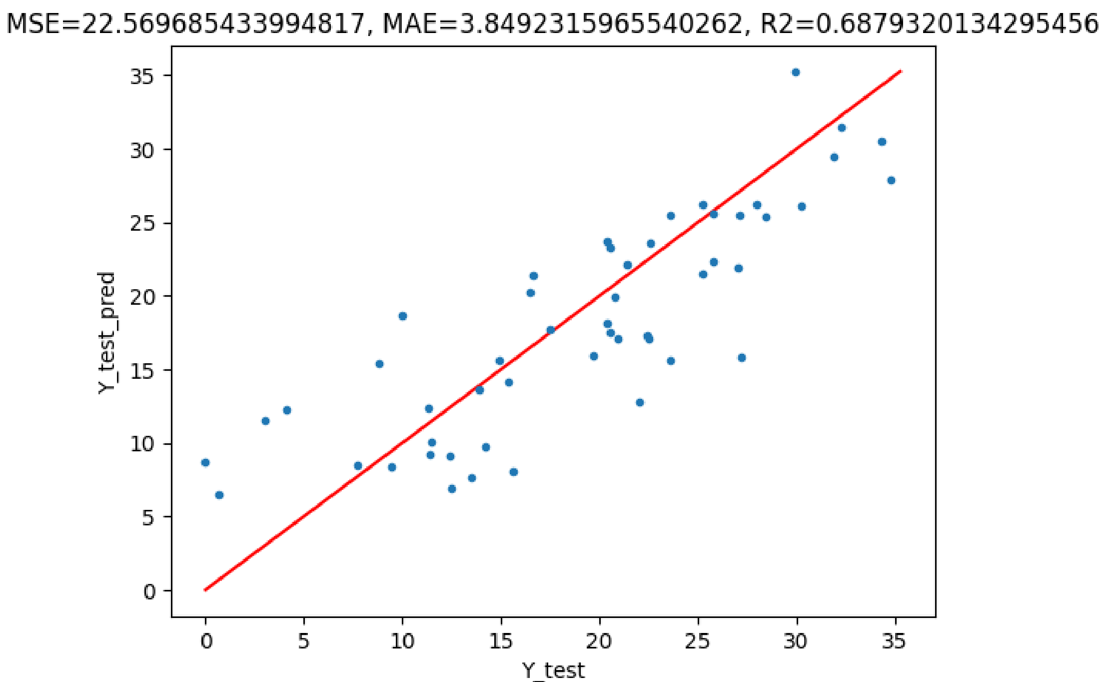


36 Testing results:

[42]:
```
MSE = np.mean((Y_test - Y_test_pred)**2)
MAE = np.mean(np.abs(Y_test - Y_test_pred))
R2 = r2_score(Y_test, Y_test_pred)
```

```
#
ymax=np.max([Y_test.max(), Y_test_pred.max()])
plt.plot(np.linspace(0,ymax, 3), np.linspace(0, ymax, 3), '-r')
plt.plot(Y_test, Y_test_pred, '.')
plt.xlabel('Y_test')
plt.ylabel('Y_test_pred')
plt.title('MSE='+str(MSE)+', MAE='+str(MAE)+', R2='+str(R2))
```

[42]:
```
Text(0.5, 1.0, 'MSE=22.569685433994817, MAE=3.8492315965540262,
R2=0.6879320134295456')
```



37 Conclusion Part 5

These models are quite exciting and have shown promising performance results. It would be really interesting to see how far we can push them on the dataset by fine-tuning their parameters. However, the only challenge is that running them can be incredibly time-consuming, and performing a grid search would take forever on my computer. Nevertheless, it is worth noting that the Category Embedding Model with the default parameters has already achieved a performance that is close to the best models we have tried so far.