Northeastern University

INFO 7390 Advances Data Science and Architecture Project on

Fitbit Dataset

Group no:- 20

Group Members:- Ankita Indi -002104957
Nirmiti Patil -002103927
Saad Ghojaria - 002129718

1. Objective:

Analyze fitbit device data to gain valuable insights on user activity.
We will aim to answer the following topics:

- Calories burnt based on daily activity
- Determine if a user is active or inactive in terms of using Fitbit and activities
- Calories burnt using DailyActivity and SleepData values

2. Background:

FitBit is a modern device used for tracking fitness and activity goals in the form of a smart wearable watch. A lot of new companies have emerged in this smartwatch market making direct connections between health and technology. These devices are wireless and help measure distance, sleep time, steps, calories and other health related data. For this project we have used the Fitbit Analysis Dataset from Kaggle.

About Dataset:
The dataset consists of 18 csv files which include information about Steps,Calories, Distance, Sleep, etc over a time period of 31 days for over 30 users.

Northeastern University

3. Introduction:

This project involves use of algorithms like K-means,Random Forest and Linear Regression aimed towards analyzing our fitbit dataset.

Libraries:- pandas,seaborn,matplotlib, numpy,plotly,sklearn, xgboost

Hyperparameter Tuning:- Elbow method for K-mean, Random Forest ,XGBoost and XGBRegressor,Linear Regression using Ridge,lasso and Elastic-Net Regression.

Feature engineering:- Correlation, Standard Scaler ,Data standardization, MinMax Scaler , ExtraTree Classifier, Principal Component Analysis, Data Transformation and Data imputation have been used throughout the project

4. Algorithms and Result:-

Random Forest

Datafiles used: DailyActivity_merged, SleepDay_merged

By plotting the correlation matrices for parameters, it is observed that most parameters are correlated. It is observed that Steps, Sedentary, total time in bed and min sleep are highly related to calories burn. Analysis is performed on the number of users logged which is 474 /936 and also for the one who did not log for a particular day.. Starting with 0 as Sunday to 6-saturday, Average steps are calculated for each day of the week along with sedentary analysis for each user which has been depicted to get a better understanding about the data.

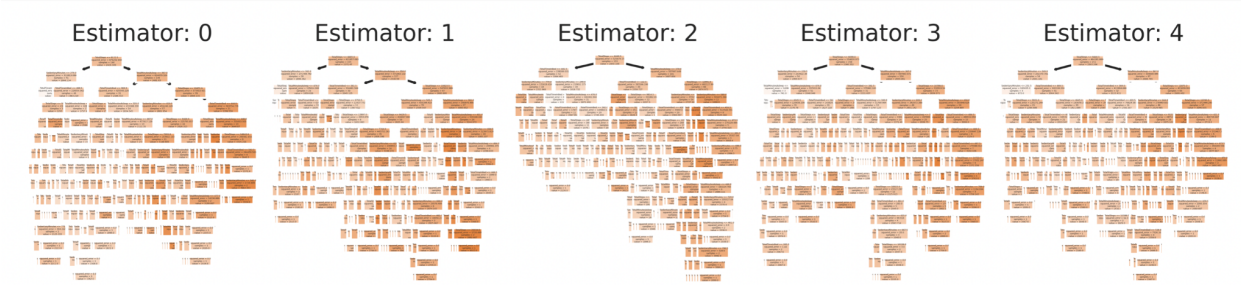
Northeastern University

```
1
rf_regressor = rf_sk(random_state=42)
rf = rf_regressor.fit(X_train, y_train)

scoring(rf, X_valid, y_valid)
```

```
Model Performance
Mean Absolute Error: 506.9816.
Mean Squared Error: 407747.2590.
R^2 Score = 0.3544.
Accuracy = 77.49%.
```

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Here are the observations for data.



XGBoost

Hyperparameter tuning is performed on XGBoost to determine the calories burnt count as Y taking 4 factors, namely SedentaryMinutes, Totalstep, TotalMinutesASleep, TotalMinutesInBed into consideration. It is observed that TotalSteps, SedentaryMinutes are directly proportional to the Calories count. SedentaryAnalysis performed on UserGroups which returns 1 if the user has average SedentaryMinutes and 0 otherwise which yields the observations as 400+ for Group 0 and 600 for Group 1.

XGBRegressor gives an accuracy of 76.06% while RF_Regressor has an accuracy of 77.49%

Northeastern University

```
## XGBoost Regressor

xgb_regressor = XGBRegressor(random_state=42)
xgb = xgb_regressor.fit(X_train, y_train)

scoring(xgb, X_valid, y_valid)

```

```
[18:35:57] WARNING: /workspace/src/objective/regression
Model Performance
Mean Absolute Error: 542.0950.
Mean Squared Error: 460604.8582.
R^2 Score = 0.2707.
Accuracy = 76.06%.
```

K-fold

K-fold is cross validation input data is divided into 'K' number of folds, hence the name K Fold. Suppose we have divided data into 5 folds i.e. K=5. Here we have 4 sets of data to train and test our model. The Models have been trained and tested 4 times, but for every iteration we will use one fold as test data and rest all as training data and here is the observation for the same.

	1st Fold	2nd Fold	3rd Fold	4th Fold	Average
LinearRegression()	0.2437	0.2679	0.3072	0.3063	0.2813
RandomForestRegressor(random_state=42)	0.4913	0.2705	0.2859	0.3394	0.3468
XGBRegressor(random_state=42)	0.4614	0.2196	0.3218	0.3095	0.3281

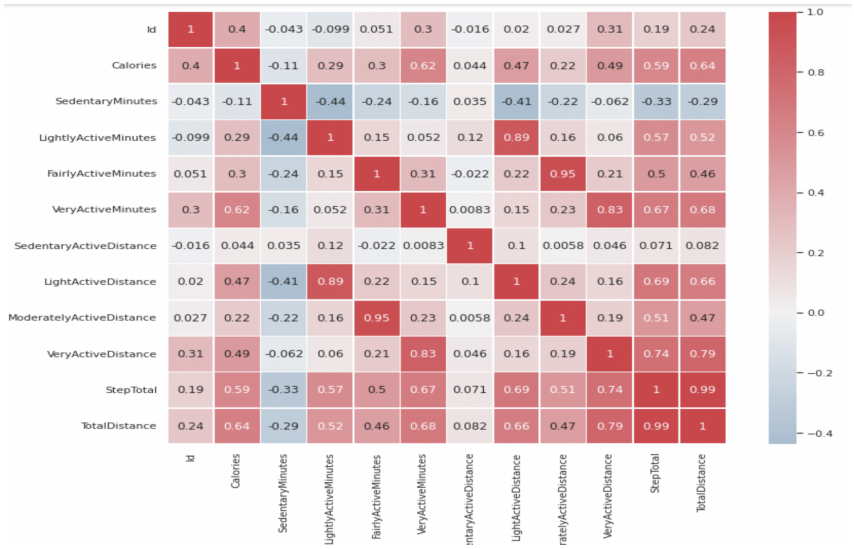
K-means

Datafiles used:- DailyActivity_merged,DailyCalories_merged,DailySteps_merged

The k-means clustering method is an unsupervised machine learning technique used to identify clusters of data objects in a dataset. Here the K-mean algorithm is used to determine whether a user is active or inactive.

Northeastern University

Feature engineering of Correlation, DataStandarization and Data Transformation has been implemented to gain the insights of the data.



Correlation observation has been performed at the values greater than 0.90 has been eliminated for further analysis.

Data transformation is the process of converting raw data into a format or structure that would be more suitable for the model or algorithm**.** 2 Data Transformation have been performed here to get better understanding.

Some of the examples we used are : -

df1['ActiveMinutes'] = df['SedentaryMinutes'] + df['LightlyActiveMinutes'] + df['FairlyActiveMinutes'] + df['VeryActiveMinutes']

df2['Distance'] = df['SedentaryActiveDistance'] + df['LightActiveDistance'] + df['VeryActiveDistance']

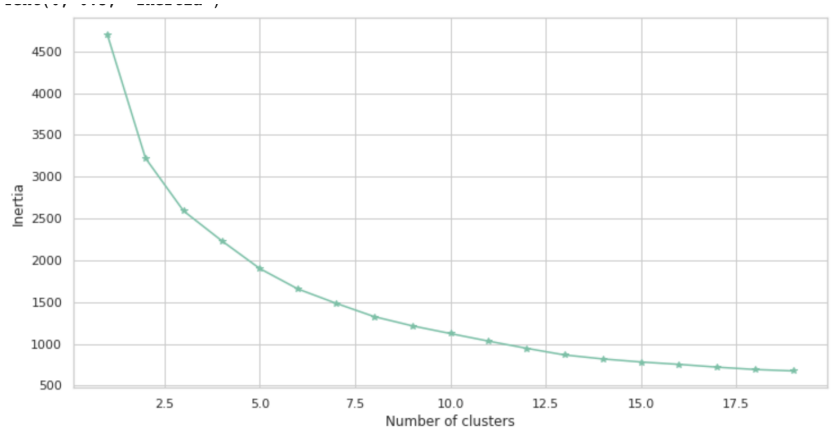
Data standardization is the process of converting data to z-score values based on the mean and standard deviation of the data .StandardScaler removes the mean and scales each feature/variable to unit variance.

Some of the examples we used are : -

After performing data cleaning, data has been standardized for feature engineering purposes. Data is classified in 2 sections, Active and Inactive.

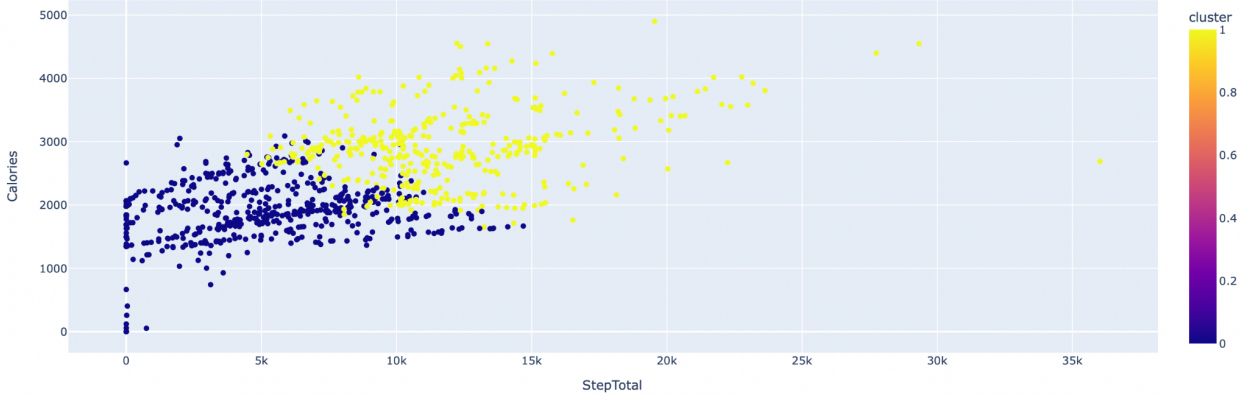
Northeastern University

To perform hyperparameter tuning, the Elbow method has been used. The SSE is defined as the sum of the squared Euclidean distances of each point to its closest centroid.According to this, it was observed that SSE distance was higher for clusters ranging from 0-2.

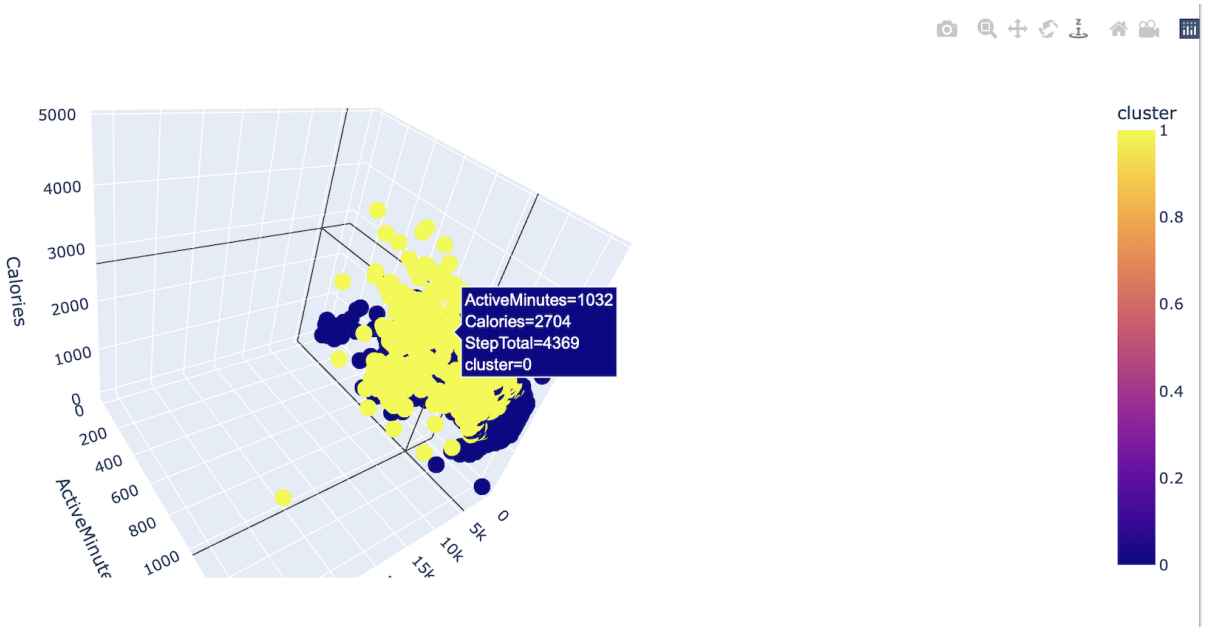


So taking the k=2 k-mean algorithm has been performed.Total sum of cluster results as 0(InActive users)-564 count
1(Active Users)-376 count.

Data Visualization has been performed taking calories and Total steps in factor.



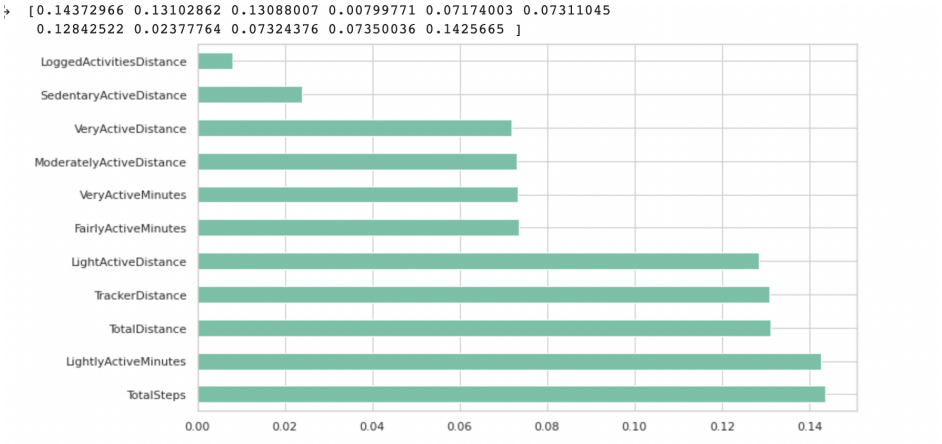
3d plot for ActiveMinutes, Calories and StepTotal



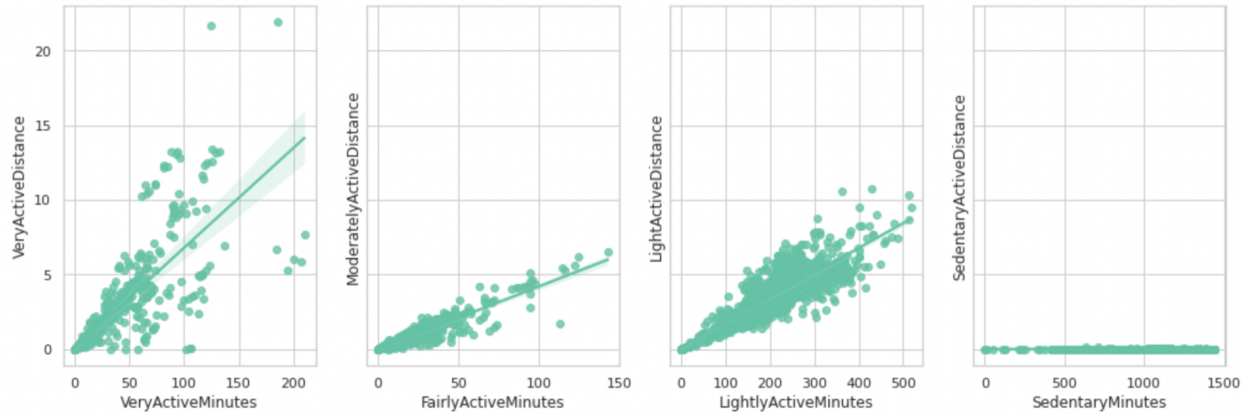
Linear Regression

Datafiles used:- DailyActivity_merged and Sleepday_merged.

Using Extra Trees Classifier to ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a "forest" to output it's classification result.

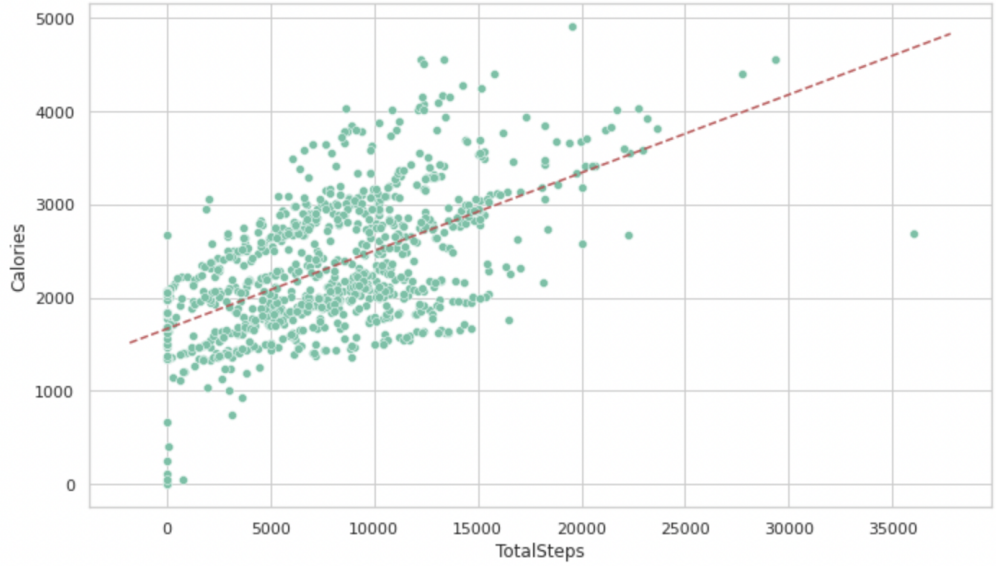


As expected, the VeryActive distances are traveled in shorter times (that is, they have larger speeds represented by steeper regression lines). A somewhat unexpected result here is that LightlyActiveMinutes lead to greater speeds than FairlyActiveMinutes. It would be interesting to know how this classification is done to actually understand the difference between "Light" activities and "Moderate" activities.



Having implemented Linear regression to fit a regression line between TotalSteps and calories

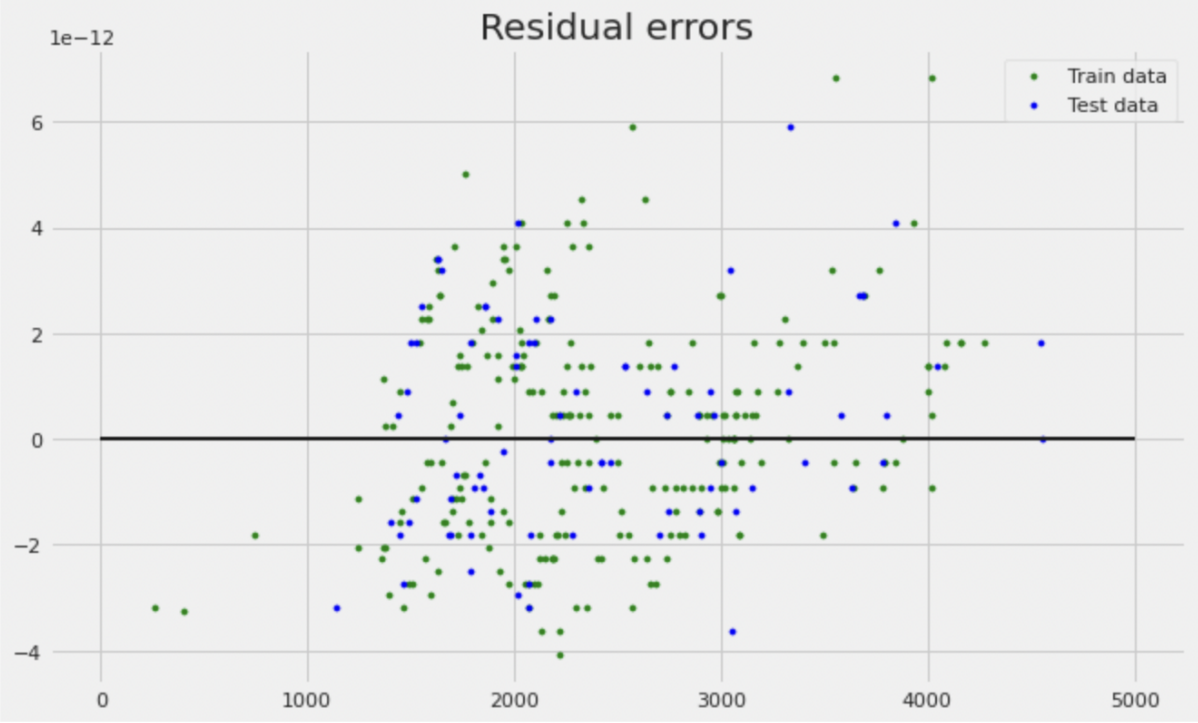
```
intercept: 1665.7426768758332
slope: [0.08351327]
```



MixMaxScalar **Transform features by scaling each feature to a given range**. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

Performing Linear Regression on DailyActivity and SleepDay data we a accuracy of accuracy has been observed

A residual is a measure of how far away a point is vertically from the regression line. Simply, it is the error between a predicted value and the observed actual value we get the following result



Hyperparameter tuning for Linear Regression:
HyperpparameterTuning is performed on Linear Regression to determine the calories burnt count as Y taking 4 factors, namely SedentaryMinutes, Totalstep, TotalMinutesASleep, TotalMinutesInBed into consideration. It is observed that TotalSteps, SedentaryMinutes are directly proportional to the Calories count.By computing the base accuracy, we have computed the relevant scores by creating the list of median predictions that has same length by fitting that function to find the Linear regression Accuracy.

Northeastern University

```
# Compute the relevant scores
base_predictions = baseline_y
base_mae = mean_absolute_error(y_valid, base_predictions)
base_mse = mean_squared_error(y_valid, base_predictions)
base_r2 = r2_score(y_valid, base_predictions)
base_errors = abs(base_predictions - y_valid)
base_mape = 100 * np.mean(base_errors / y_valid)
base_accuracy = 100 - base_mape
print('Model Performance')
print('Mean Absolute Error: {:0.4f}.'.format(base_mae))
print('Mean Squared Error: {:0.4f}.'.format(base_mse))
print('R^2 Score = {:0.4f}.'.format(base_r2))
print('Accuracy = {:0.2f}%.'.format(base_accuracy))
```

```
Model Performance
Mean Absolute Error: 649.0120.
Mean Squared Error: 650005.8072.
R^2 Score = -0.0292.
Accuracy = 72.74%.
```

In [65]:
```
regressor = LinearRegression()
mlr = regressor.fit(X_train, y_train)

scoring(mlr, X_valid, y_valid)
```

```
Model Performance
Mean Absolute Error: 571.5923.
Mean Squared Error: 461353.1211.
R^2 Score = 0.2695.
Accuracy = 74.79%.
```

Linear Regression gives Accuracy of 74.79%

Now, I would be Hyper tuning these models using ridge,lasso and Elastic-Net Regression.

Hypertuning using Ridge:

In [96]:
```
# Train model with default alpha=1
ridge = Ridge(alpha=10000).fit(X_train, y_train)
# get cross val scores
scoring(ridge, X_valid, y_valid)
```

```
Model Performance
Mean Absolute Error: 573.7368.
Mean Squared Error: 463473.5759.
R^2 Score = 0.2662.
Accuracy = 74.66%.
```

Accuracy :74.66%

Hypertuning using Elastic-Net Regression:

In [101]:
```
# Train model with default alpha=1
elastic_net = ElasticNet(alpha=70, l1_ratio=0.3).fit(X_train, y_train)
# get cross val scores
scoring(elastic_net, X_valid, y_valid)
```

```
Model Performance
Mean Absolute Error: 574.3572.
Mean Squared Error: 464119.3491.
R^2 Score = 0.2651.
Accuracy = 74.63%.
```

Accuracy: 74.63%

Northeastern University

Hypertuning using lasso:

```
In [93]: # Train model with default alpha=1
         lasso = Lasso(alpha=100).fit(X_train, y_train)
         # get cross val scores
         scoring(lasso, X_valid, y_valid)
```

```
Model Performance
Mean Absolute Error: 572.5824.
Mean Squared Error: 462261.6331.
R^2 Score = 0.2681.
Accuracy = 74.73%.
```

Accuracy:74.73%

Comparing Accuracy of all the Hypertunning models , the Accuracy of the Linear regression remains best fit with Accuracy:74.79%