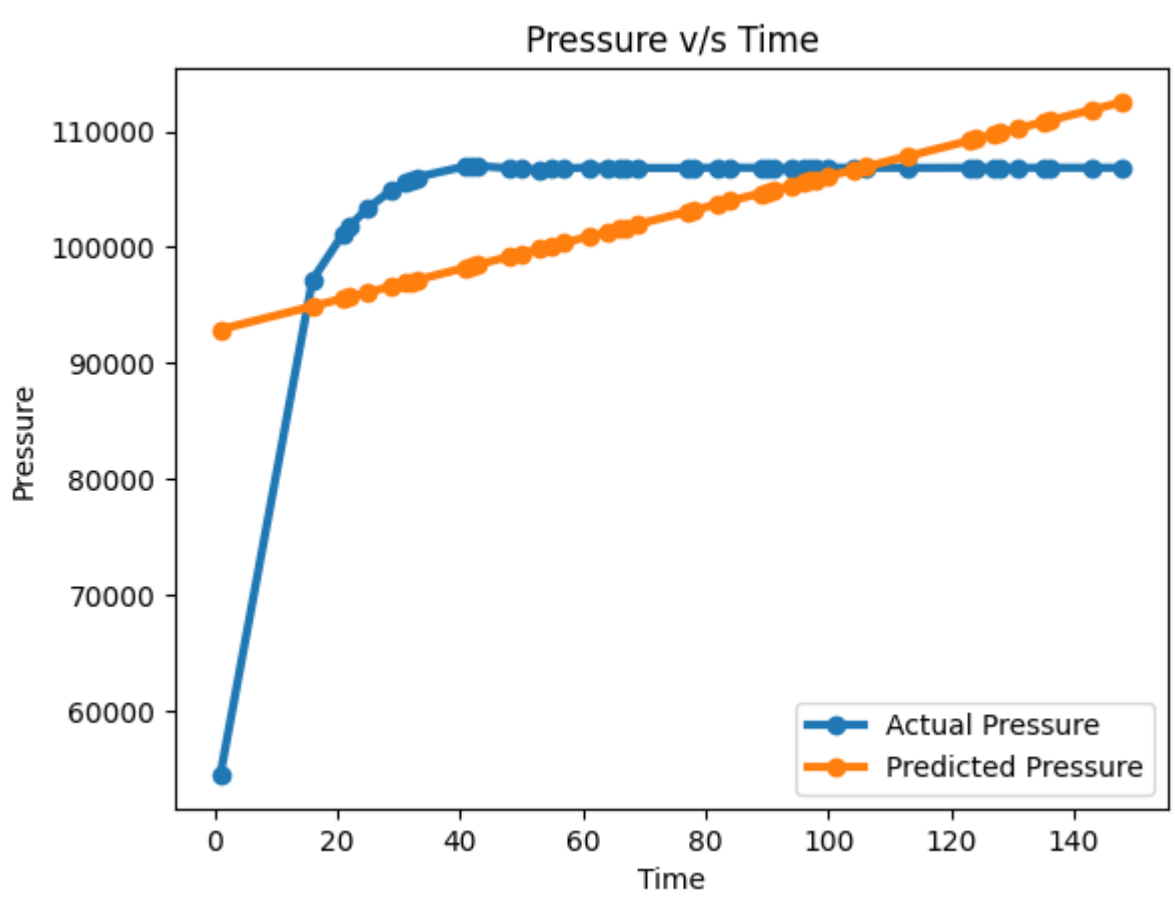
Machine Learning Models

file_name: " P_vs_T "

Select_ML_Model: LINEAR REGRESSION PREDICTION

plot_only_graph: ☑

Show code




```
mae:  5231.659798136245

mse:  58684684.27086035

r2_score:  0.02175338776736957
```

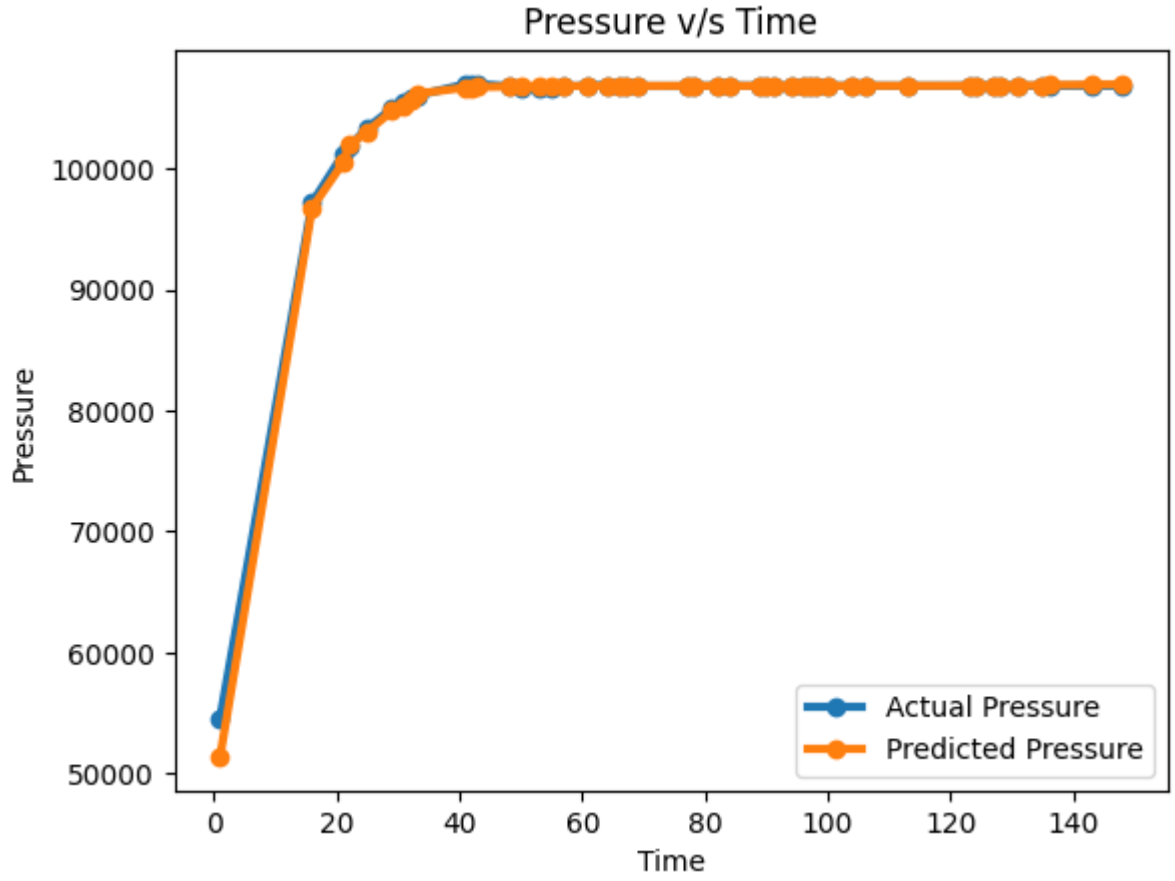
Machine Learning Models

file_name: " P_vs_T "

Select_ML_Model: RANDOM FOREST REGRESSION

plot_only_graph: ☑

Show code




```
mae:  129.71506065218145

mse:  244322.75860807922

r2_score:  0.9959272523339048
```

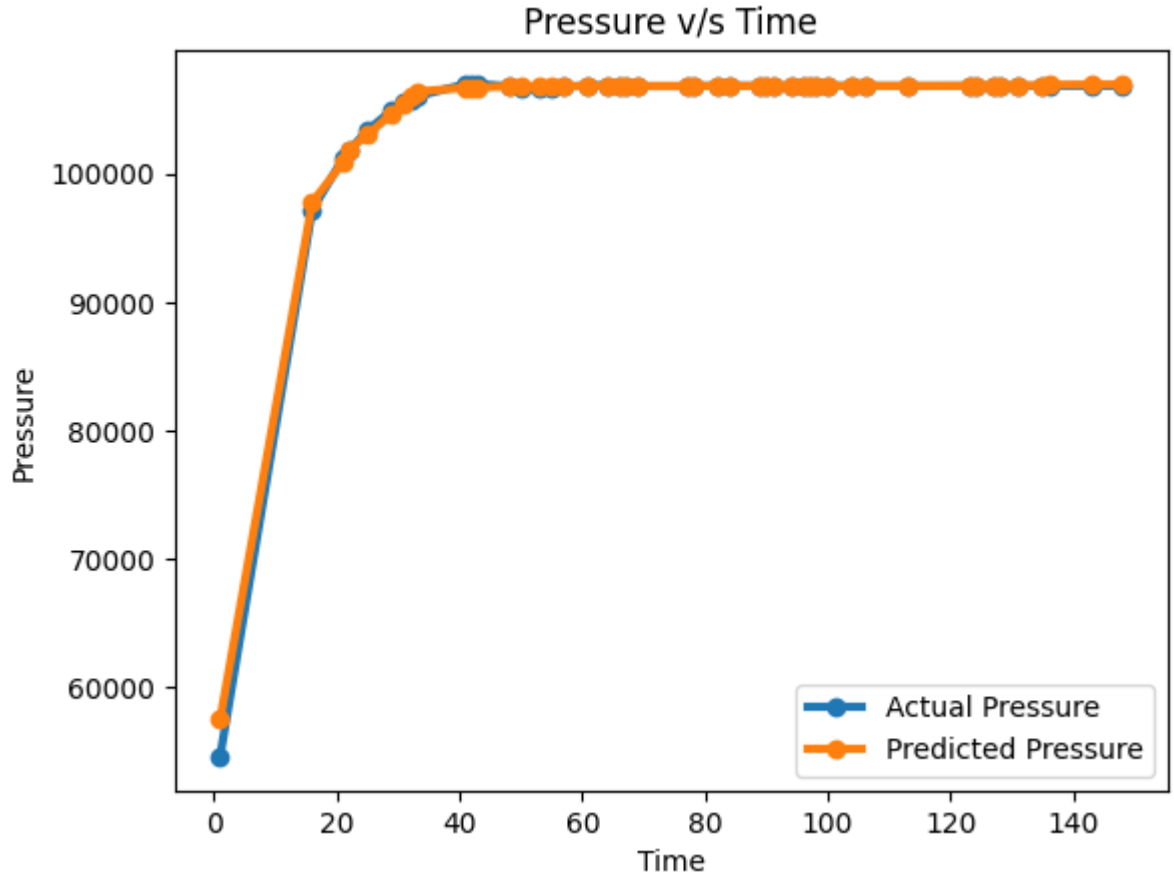
Machine Learning Models

file_name: " P_vs_T "

Select_ML_Model: KNN REGRESSION

plot_only_graph: ☑

Show code




```
mae:  119.31876021739235

mse:  198497.74740654047

r2_score:  0.9966911341289659
```

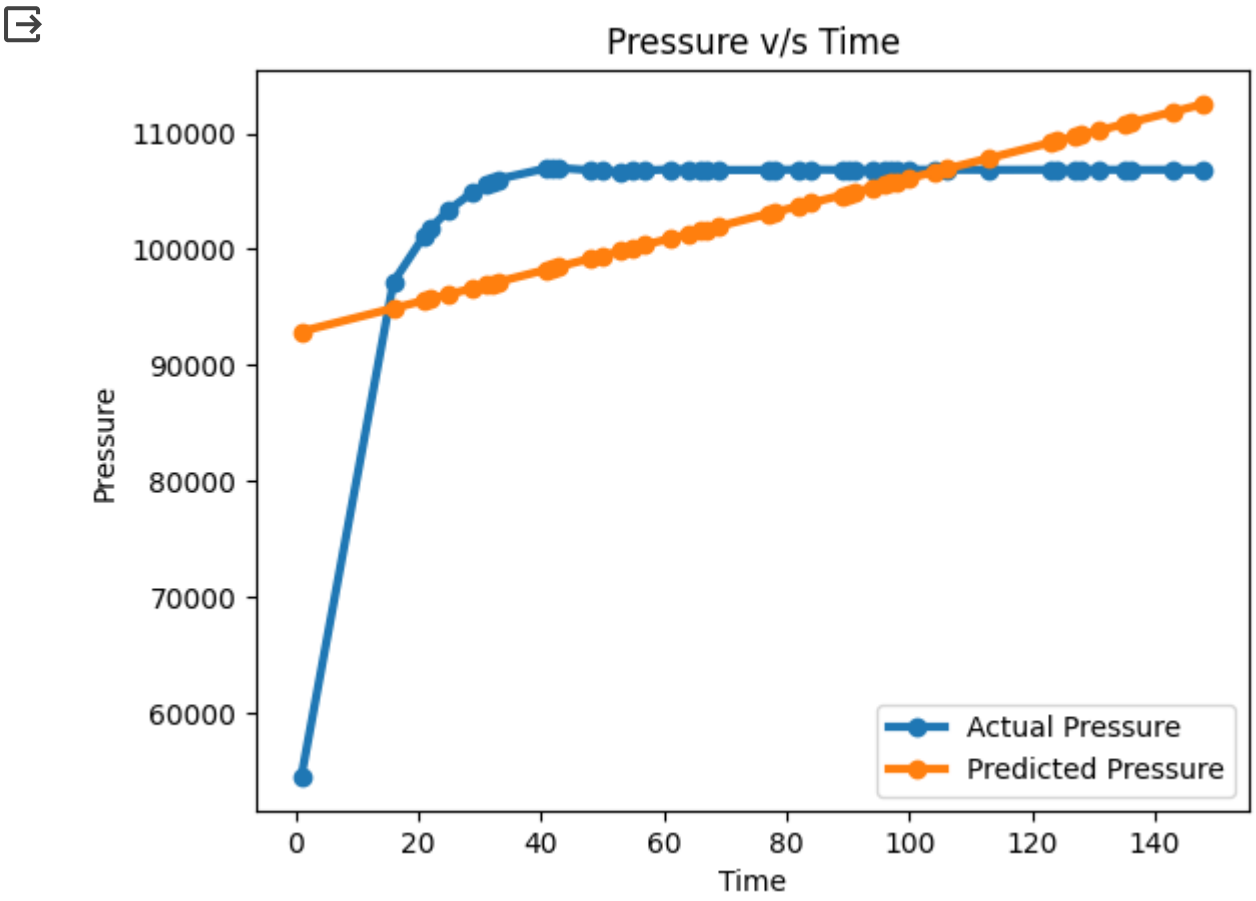
Machine Learning Models

file_name: " P_vs_T "

Select_ML_Model: LASSO REGRESSION

plot_only_graph: ☑

Show code




```
mae:  5231.649924460145

mse:  58684614.21448919

r2_score:  0.021754555574755807
```

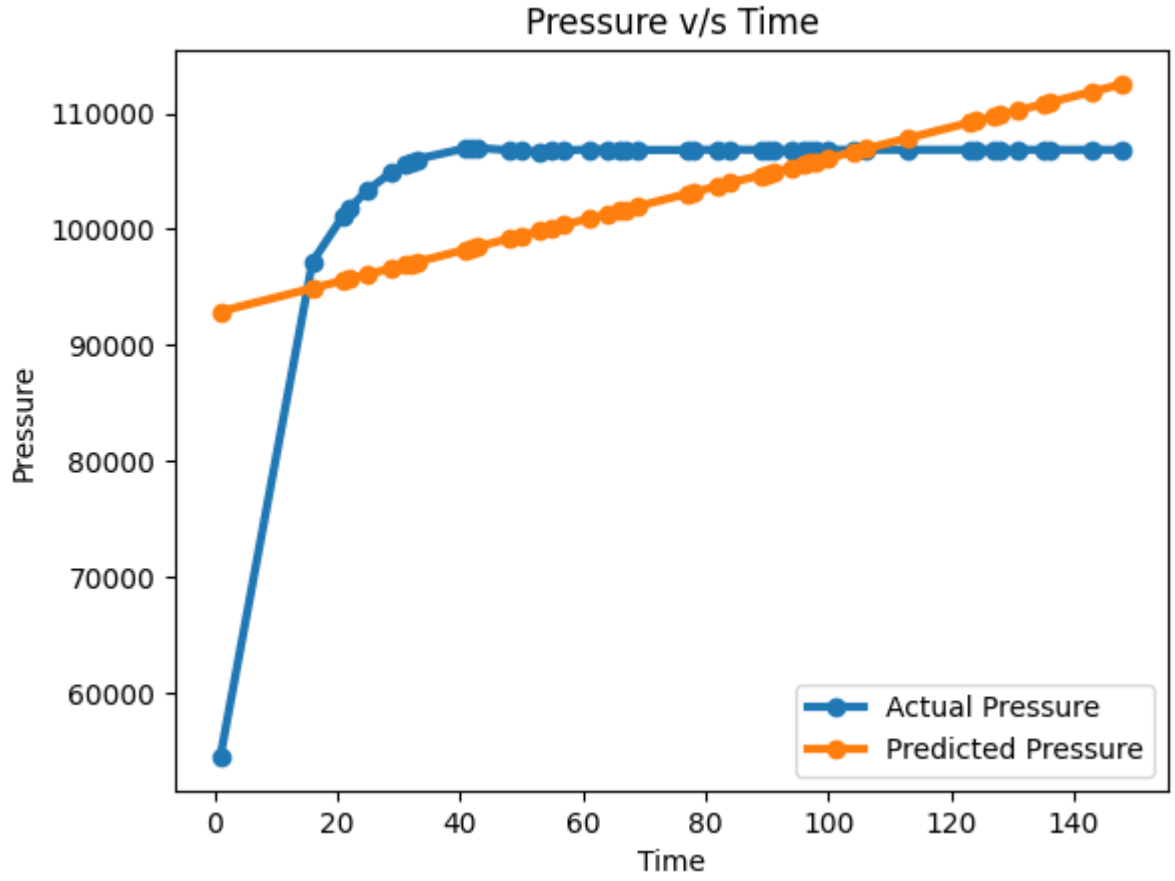
Machine Learning Models

file_name: " P_vs_T "

Select_ML_Model: POLYNOMIAL REGRESSION

plot_only_graph: ☑

Show code




```
mae:  5231.659798136245

mse:  58684684.27086035

r2_score:  0.02175338776736957
```

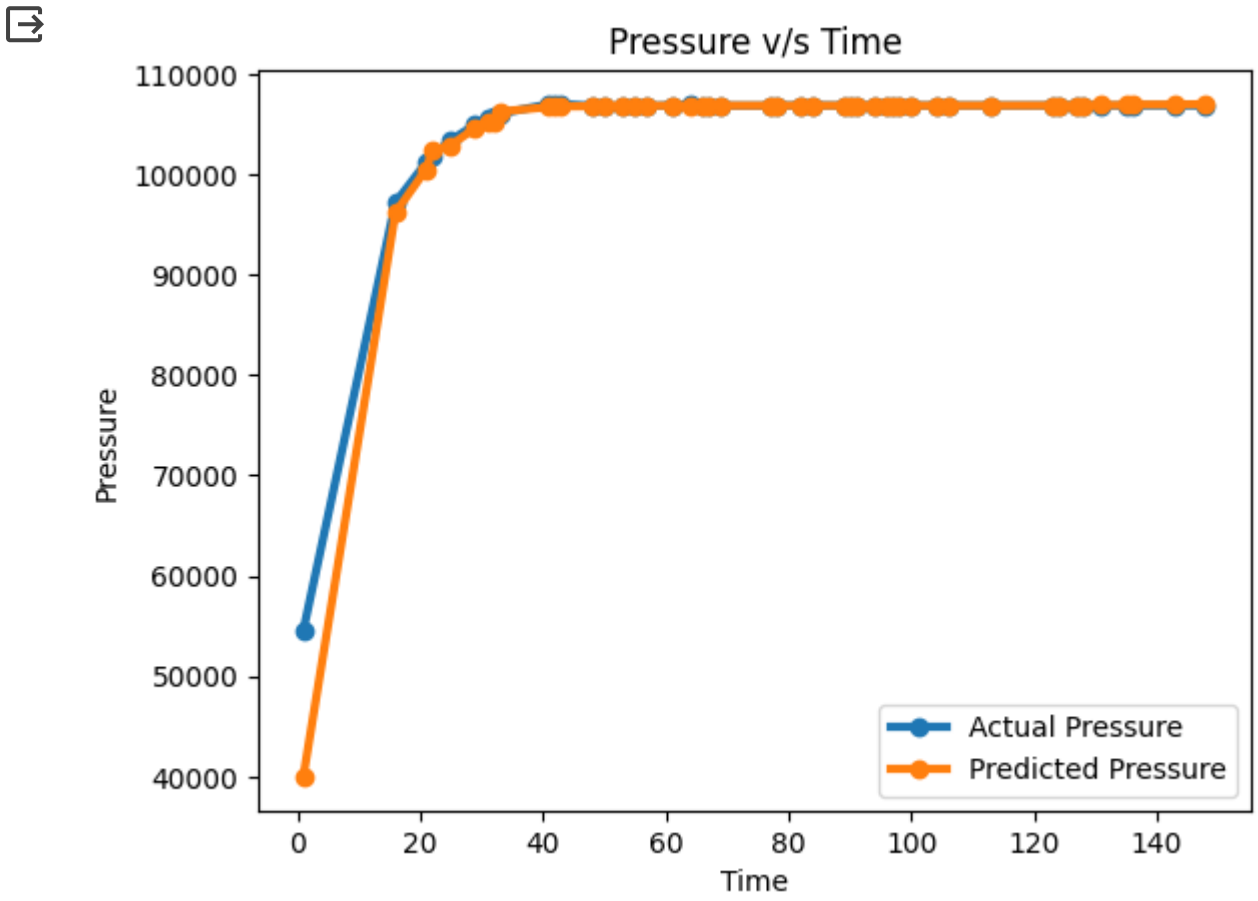
Machine Learning Models

file_name: " P_vs_T "

Select_ML_Model: DECISION TREE REGRESSION

plot_only_graph: ☑

Show code




```
mae:  414.2929969565204

mse:  4672508.859216338

r2_score:  0.9221114330093622
```

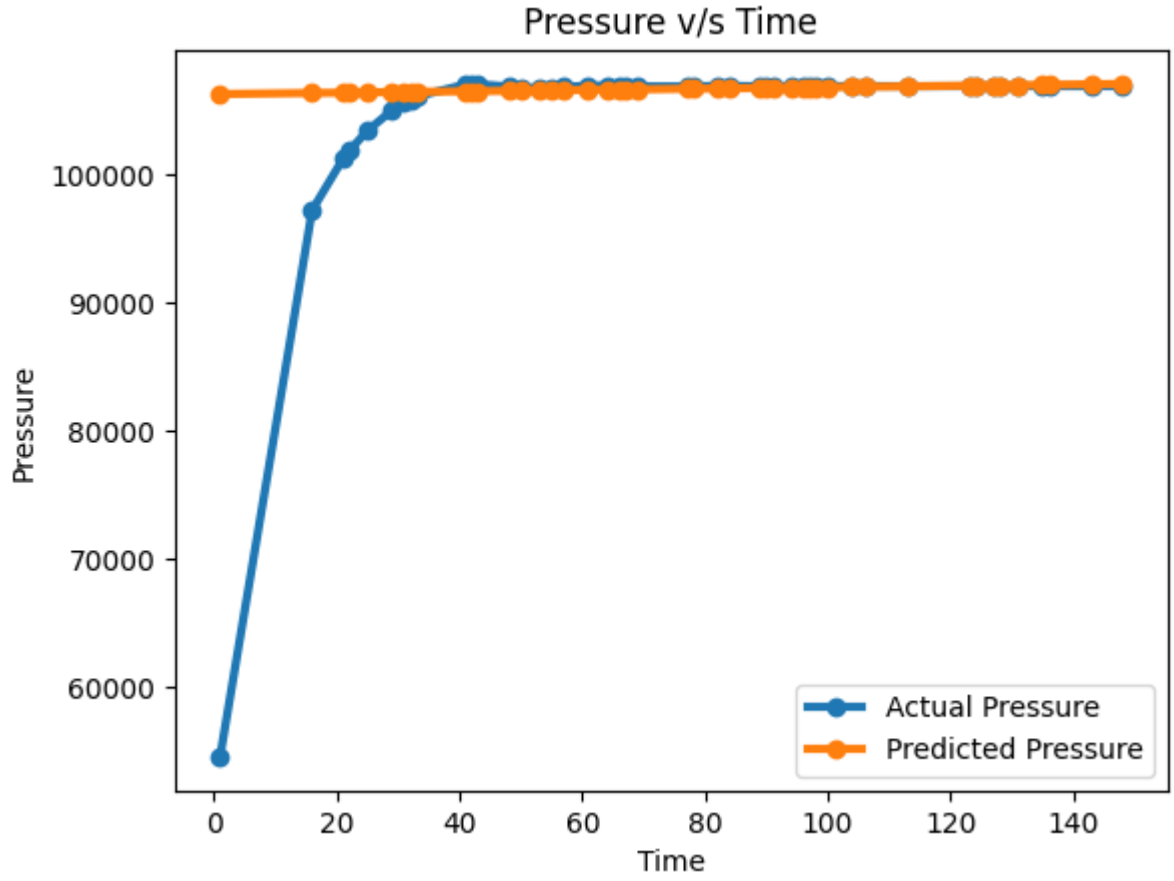
Machine Learning Models

file_name: " P_vs_T "

Select_ML_Model: SUPPORT VECTOR REGRESSION

plot_only_graph: ☑

Show code




```
mae:  1816.9366932609214

mse:  61164016.47874037

r2_score:  -0.01957593628187304
```

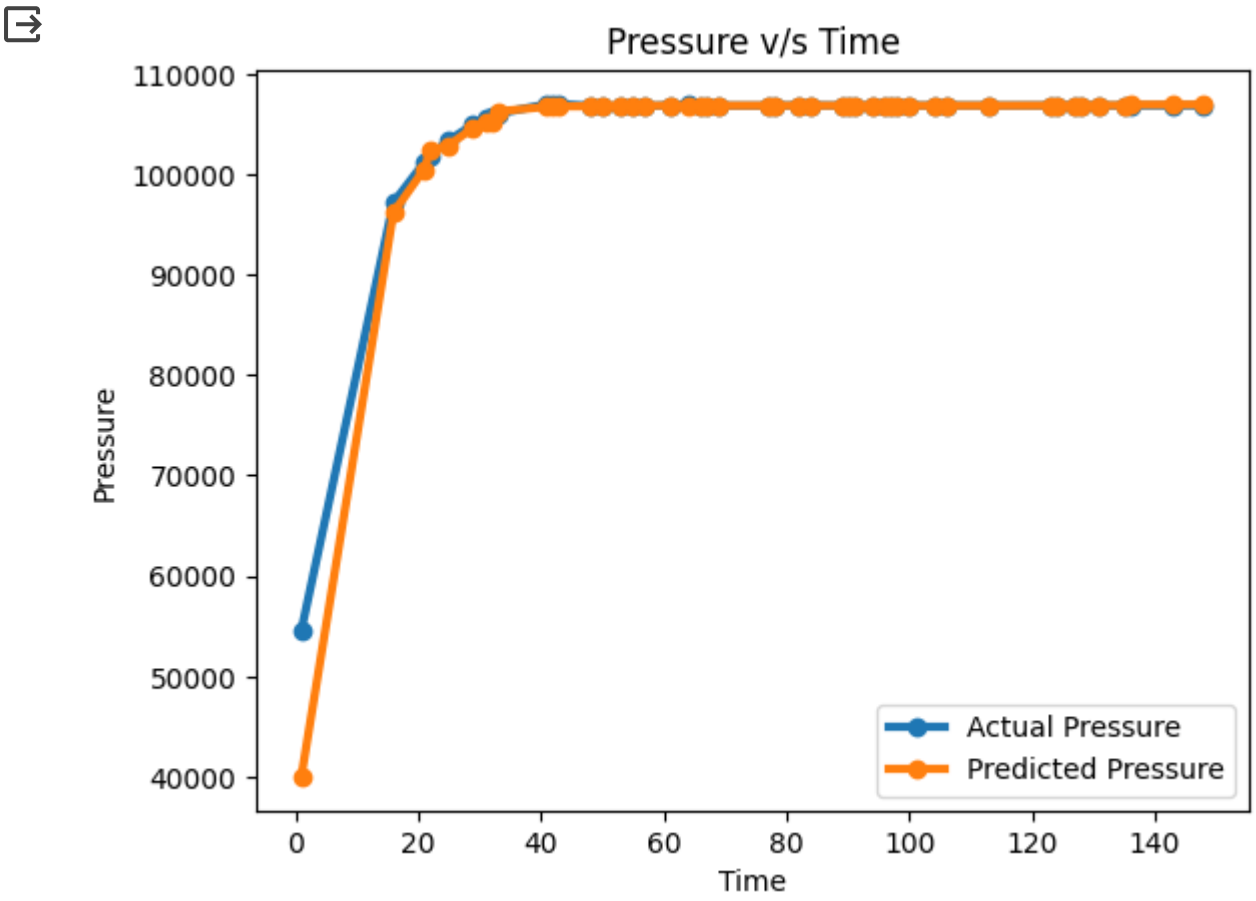
Machine Learning Models

file_name: " P_vs_T "

Select_ML_Model: GRADIENT BOOSTING REGRESSOR

plot_only_graph: ☑

Show code




```
mae:  417.09723906876513

mse:  4670127.254252905

r2_score:  0.9221511332653336
```

Name: Vatsal Arya

Roll No.: 12

Machine Learning Models

```
#@title Machine Learning Models
# Import important libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import warnings
warnings.filterwarnings('ignore')

file_name = "P_vs_T" #@param {type:"string"}
data = pd.read_csv(file_name+'.csv')
df = list(data.columns)
data.corr()

#Model Preparation:
X = data[[df[0]]]
y = data[df[1]]
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=101)

Select_ML_Model = 'ELASTICNET REGRESSION' #@param ["LINEAR REGRESSION PREDICTION","RANDO
mlm = Select_ML_Model
if mlm == 'LINEAR REGRESSION PREDICTION':
  from sklearn.linear_model import LinearRegression
  reg = LinearRegression()
elif mlm == 'RANDOM FOREST REGRESSION':
  from sklearn.ensemble import RandomForestRegressor
  reg = RandomForestRegressor(n_estimators=50,max_depth=30,n_jobs=-1)
elif mlm == 'KNN REGRESSION':
  from sklearn.neighbors import KNeighborsRegressor
  reg = KNeighborsRegressor(n_neighbors=3)
elif mlm == 'LASSO REGRESSION':
  from sklearn.linear_model import Lasso
  reg = Lasso()
elif mlm == 'POLYNOMIAL REGRESSION':
  from sklearn.preprocessing import PolynomialFeatures
  polymatrix = PolynomialFeatures(degree=4)
  x_poly = polymatrix.fit_transform(X_train)
  reg = LinearRegression()
elif mlm == 'DECISION TREE REGRESSION':
  from sklearn.tree import DecisionTreeRegressor
  reg = DecisionTreeRegressor(random_state=0)
elif mlm == 'SUPPORT VECTOR REGRESSION':
  from sklearn.svm import SVR
  reg = SVR(C=100,kernel='linear',gamma=1)
elif mlm == 'GRADIENT BOOSTING REGRESSOR':
```

file_name: " P_vs_T "

Select_ML_Model: ELASTICNET REGRI

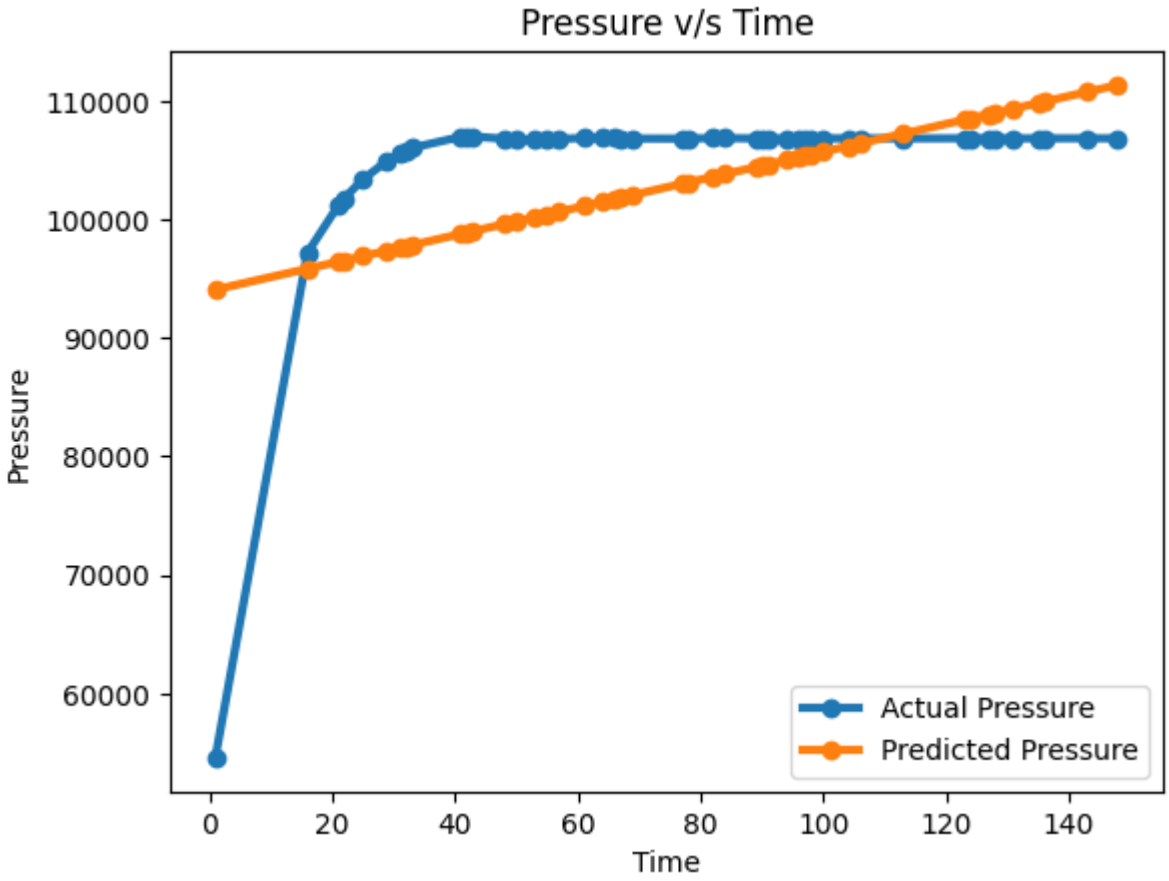
plot_only_graph: ☑

```
    from sklearn.ensemble import GradientBoostingRegressor
    reg = GradientBoostingRegressor(learning_rate=0.1,random_state=4)
  elif mlm == 'LGBM REGRESSOR':
    from lightgbm import LGBMRegressor
    reg = LGBMRegressor(learning_rate=0.1,random_state=30)
  elif mlm == 'ELASTICNET REGRESSION':
    from sklearn.linear_model import ElasticNetCV
    reg = ElasticNetCV(cv=10).fit(X_train,y_train)
  else:
    print('Select a ML Model')

  reg.fit(X_train,y_train)
  predictions = reg.predict(X_test)
  data = pd.DataFrame({'Actual Values':y_test,'Predicted Values':predictions})
  data.sort_index(ascending=True)
  plot_only_graph = True #@param {type:"boolean"}
  if plot_only_graph == False:
    print('\n',data)

  yrs_exp = X_test.sort_index(ascending=True)
  sal_act = y_test.sort_index(ascending=True)
  sal_pre = np.sort(predictions)

  plt.plot(yrs_exp, sal_act, label = 'Actual '+df[1], marker = "o", linewidth=3)
  plt.plot(yrs_exp, sal_pre, label = 'Predicted '+df[1], marker = "o", linewidth=3)
  plt.xlabel(df[0])
  plt.ylabel(df[1])
  plt.title(df[1]+' v/s '+df[0])
  plt.legend(loc='lower right')
  plt.show()
  from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
  mae = print('mae: ',mean_absolute_error(y_test,predictions))
  mse = print('\nmse: ',mean_squared_error(y_test,predictions))
  rscore = print('\nr2_score: ',r2_score(y_test,predictions))
```




```
mae:  4925.850322284051

mse:  56752732.55025215

r2_score:  0.0539581316313561
```