# Appendix-1
# Coding for prediction of straightness by Machine Learning Algorithm

```
# Coding for Bar Straightening Prediction by Machine Learning approaches

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time, math
Al = pd.read_csv("Al_after.csv")
Al.describe()
X = Al.drop(columns=['Deflection (in mm)'])
y = Al['Deflection (in mm)']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

## Linear Regression……..

from sklearn import linear_model, metrics
reg = linear_model.LinearRegression(fit_intercept=True, copy_X=True, n_jobs=None)
start_time = time.time()
reg.fit(X_train, y_train)
training_time = time.time() - start_time
print("Model training time: %.5f seconds" % training_time)
start_time = time.time()
y_pred_LR=reg.predict(X_test)
prediction_time = time.time()print("Prediction time: %.5f seconds" % prediction_time)
print("Quantitative performance status of LR model :- ")
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
print("R2_square_LR model value is : %.5f"% (r2_score(y_test, y_pred_LR)))
print("MAE_LR model value is : %.5f"% (mean_absolute_error(y_test, y_pred_LR)))
print("MSE_LR model value is : %.5f"% (mean_squared_error(y_test, y_pred_LR)))
rmseLR = math.sqrt(mean_squared_error(y_test, y_pred_LR))
print("RMSE_LR model value is: %.5f"% rmseLR)
```

```
# LR Prediction plot.....

import matplotlib.pyplot as plt
plt.figure(figsize=(5,3))
df_pred_LR = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_LR})
plt.plot(df_pred_LR.index, df_pred_LR['Actual'], label='Actual', color='blue', marker='o', linestyle='None',
markersize=3)
plt.plot(df_pred_LR.index, df_pred_LR['Predicted'], label='Predicted', color='red', marker='s', linestyle='None',
markersize=3)
plt.xlabel('Index')
plt.ylabel('Deflections (x 0.01 mm)')
plt.title('LR Prediction Plot of Al after straightening')
plt.legend()
plt.grid(True)
plt.show()

# Create the LR residual plot.............

import matplotlib.pyplot as plt
import numpy as np
residuals_LR = y_test - y_pred_LR
plt.figure(figsize=(5, 3))
plt.scatter(y_pred_LR, residuals_LR, s=10)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlaplt.ylabel('Residuals')
plt.title('LR Residual Plot of Al after straightening')
plt.show()

# Random Forest Regressor…….

from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(max_depth = 10, min_samples_leaf = 1, min_samples_split = 2, n_estimators =
200)
start_time = time.time()
rfr.fit(X_train,y_train)
training_time = time.time() - start_time
print("Model training time: %.5f seconds" % training_time)
```

```
start_time = time.time()
y_pred_rfr = rfr.predict(X_test)
prediction_time = time.time() - start_time
print("Prediction time: %.5f seconds" % prediction_time)
print("Quantitative performance status of RFR model :- ")
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import math
print("R2_square_RFR model value is : %.5f"% (r2_score(y_test, y_pred_rfr)))
print("MAE_RFR model value is : %.5f"% (mean_absolute_error(y_test, y_pred_rfr)))
print("MSE_RFR model value is : %.5f"% (mean_squared_error(y_test, y_pred_rfr)))
rmse_rfr = math.sqrt(mean_squared_error(y_test, y_pred_rfr))
print("RMSE_RFR model value is: %.5f"% rmse_rfr)

# RFR Prediction plot.....
import matplotlib.pyplot as plt
plt.figure(figsize=(5,3))
df_pred_rfr = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_rfr})
plt.plot(df_pred_rfr.index, df_pred_rfr['Actual'], label='Actual', color='blue', marker='o', linestyle='None',
markersize=3)
plt.plot(df_pred_rfr.index, df_pred_rfr['Predicted'], label='Predicted', color='red', marker='s', linestyle='None',
markersize=3)plt.xlabel('Index')
plt.ylabel('Deflections (x 0.01 mm)')
plt.title('RFR Prediction Plot of 10 mm Al after straightening')
plt.legend()
plt.show()

# Create the RFR residual plot.....
import matplotlib.pyplot as plt
import numpy as np
residuals_rfr = y_test - y_pred_rfr
plt.figure(figsize=(5, 3))
plt.scatter(y_pred_rfr, residuals_rfr, s=10)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('RFR Residual Plot of Al after straightening')
plt.figure(figsize=(5,3))
plt.show()
```

```
# Support Vector Regressor…….
from sklearn import svm
svm = svm.SVR(C=1.0, epsilon=0.1, kernel='rbf', gamma='scale')
start_time = time.time()
svm.fit(X_train, y_train)
training_time = time.time() - start_time
print("Model training time: %.5f seconds" % training_time)
start_time = time.time()
y_pred_svm = svm.predict(X_test)
prediction_time = time.time() - start_time
print("Prediction time: %.5f seconds" % prediction_time)
print("Quantitative performance status of SVR model :from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error,
import math
print("R2_square_SVR model value is : %.5f"% (r2_score(y_test, y_pred_svm)))
print("MAE_SVR model value is : %.5f"% (mean_absolute_error(y_test, y_pred_svm)))
print("MSE_SVR model value is : %.5f"% (mean_squared_error(y_test, y_pred_svm)))
rmse_svm = math.sqrt(mean_squared_error(y_test, y_pred_svm))
print("RMSE_SVR model value is: %.5f"% rmse_svm)

# SVR Prediction plot.....
import matplotlib.pyplot as plt
plt.figure(figsize=(5,3))
df_pred_svm = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_svm})
plt.plot(df_pred_svm.index, df_pred_svm['Actual'], label='Actual', color='blue', marker='o', linestyle='None',
markersize=3)
plt.plot(df_pred_svm.index, df_pred_svm['Predicted'], label='Predicted', color='red', marker='s', linestyle='None', markersize=3)
plt.xlabel('Index')
plt.ylabel('Deflections (x 0.01 mm)')
plt.title('SVR Prediction Plot of Al after straightening')
plt.legend()
plt.show()

# Create the SVR residual plot.....
import matplotlib.pyplot as plt
import numpy as np
residuals_svm = y_test - y_pred_svm
```

```
plt.figure(figsize=(5, 3))
plt.scatter(y_pred_svm, residuals_svm, s=10)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('SVR Residual Plot of Al after straightening')
plt.figure(figsize=(5,3))
plt.show()


## Extreme Gradient Boost Regressor (XGB)…….
from xgboost import XGBRegressor
xgb = XGBRegressor(
max_depth=5,
learning_rate=0.1,
n_estimators=200,
gamma=0,
subsample=0.8,
colsample_bytree=0.8,
reg_alpha=0.1,
reg_lambda=1,
n_jobs=-1
)
start_time = time.time()
xgb.fit(X_train, y_train)
training_time = time.time() - start_time
print("Model training time: %.5f seconds" % training_time)
start_time = time.time()
y_pred_xgb = xgb.predict(X_test)
prediction_time = time.time() - start_time
print("Prediction time: %.5f seconds" % prediction_time)
print("Quantitative performance status of XGBoost model :- ")
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import math
print("R2_square_XGB model value is : %.5f"% (r2_score(y_test, y_pred_xgb)))
print("MAE_XGB model value is : %.5f"% (mean_absolute_error(y_test, y_pred_xgb)))
print("MSE_XGB model value is : %.5f"% (mean_squared_error(y_test, y_pred_xgb)))
rmse_xgb = math.sqrt(mean_squared_error(y_test, y_pred_xgb))
print("RMSE_XGB model value is: %.5f"% rmse_xgb) # XGBoost Prediction plot in same x-
axis .....
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(5,3))
df_pred_xgb = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_xgb})
plt.plot(df_pred_xgb.index, df_pred_xgb['Actual'], label='Actual', color='blue', marker='o', linestyle='None',
markersize=3)
plt.plot(df_pred_xgb.index, df_pred_xgb['Predicted'], label='Predicted', color='red', marker='s', linestyle='None',
markersize=3)
plt.xlabel('Index')
plt.ylabel('Deflections (x 0.01 mm)')
plt.title('XGB Prediction Plot of Al after straightening')
plt.legend()
plt.show()

# Create the XGBoost residual plot.....
import matplotlib.pyplot as plt
import numpy as np
residuals_xgb = y_test - y_pred_xgb
plt.figure(figsize=(5, 3))
plt.scatter(y_pred_xgb, residuals_xgb, s=10)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('XGB Residual Plot of Al after straightening')
plt.figure(figsize=(5,3))
plt.show()

# Decision Tree Regressor……
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor(max_depth=5, min_samples_split=2, min_samples_leaf=1)
start_time = time.time()
dtr.fit(X_train,y_train)training_time = time.time() - start_time
print("Model training time: %.5f seconds" % training_time)
start_time = time.time()
y_pred_dtr = dtr.predict(X_test)
prediction_time = time.time() - start_time
print("Prediction time: %.5f seconds" % prediction_time)
print("Quantitative performance status of Decision Tree model :- ")
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

```
import math
print("R2_square_DTR model value is : %.5f"% (r2_score(y_test, y_pred_dtr)))
print("MAE_DTR model value is : %.5f"% (mean_absolute_error(y_test, y_pred_dtr)))
print("MSE_DTR model value is : %.5f"% (mean_squared_error(y_test, y_pred_dtr)))
rmse_dtr = math.sqrt(mean_squared_error(y_test, y_pred_dtr))
print("RMSE_DTR model value is: %.5f"% rmse_dtr)

# Decision Tree Prediction plot …….....
import matplotlib.pyplot as plt
plt.figure(figsize=(5,3))
df_pred_dtr = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_dtr})
plt.plot(df_pred_dtr.index, df_pred_dtr['Actual'], label='Actual', color='blue', marker='o', linestyle='None',
markersize=3)
plt.plot(df_pred_dtr.index, df_pred_dtr['Predicted'], label='Predicted', color='red', marker='s', linestyle='None',
markersize=3)
plt.xlabel('Index')
plt.ylabel('Deflections (x 0.01 mm)')
plt.title('Decision Tree Prediction Plot of Al after straightening')
plt.legend()
plt.show()

# Create the Decision Tree residual plot.....
importimport numpy as np
residuals_dtr = y_test - y_pred_dtr
plt.figure(figsize=(5, 3))
plt.scatter(y_pred_dtr, residuals_dtr, s=10)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Decision Tree Residual Plot of Al after straightening')
plt.figure(figsize=(5,3))
plt.show()
```