```
from types import CodeType

from typing import Tuple, Type

from matplotlib import style

from matplotlib.markers import MarkerStyle

from numpy.core.defchararray import title

from numpy.core.fromnumeric import ravel, size

from seaborn import colors

from seaborn.palettes import SEABORN_PALETTES, color_palette

from operator import index

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from seaborn.utils import saturate

from sklearn import preprocessing, utils

from sklearn.svm import SVR

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear_model import LinearRegression,Ridge,Lasso,ElasticNet

from sklearn import neighbors,tree

from sklearn.model_selection import train_test_split,StratifiedKFold

from sklearn.metrics import mean_squared_error, r2_score

from sklearn import preprocessing as per

from sklearn.preprocessing import StandardScaler, Normalizer

from sklearn.utils.extmath import weighted_mode

################################################################################
```

```
# To Create All of Regression Models

def Multi_RegRession(X_train:np.ndarray,y_train:np.ndarray,X_test:np.ndarray,y_test:np.ndarray,type:str):

    if type == "l":

        Regression = LinearRegression().fit(X_train,y_train)

        predict = Regression.predict(X_test)

    elif type == "R":

        Regression = Ridge().fit(X_train,y_train)

        predict = Regression.predict(X_test)

    elif type == "L":

        Regression = Lasso().fit(X_train,y_train)

        predict = Regression.predict(X_test)

    elif type == "E":

        Regression = ElasticNet().fit(X_train,y_train)

        predict = Regression.predict(X_test)

    elif type == "K":

        Regression = neighbors.KNeighborsClassifier().fit(X_train,ravel(y_train))

        predict = Regression.predict(X_test)

    elif type == "D":

        Regression = tree.DecisionTreeClassifier().fit(X_train,y_train)

        predict = Regression.predict(X_test)

    elif type == "RF":

        Regression = RandomForestClassifier().fit(X_train,ravel(y_train))

        predict = Regression.predict(X_test)

    elif type == "S":
```

```
        Regression = SVR().fit(X_train,ravel(y_train))

        predict = Regression.predict(X_test)

    else: print(f"{type} not found in Regression typies")



    r2 = r2_score(y_test,predict)

    Mse = mean_squared_error(y_test,predict)



    return Regression, predict, r2, Mse



# To Draw Residual Analusis

def resudual_analysis(y_test:np.ndarray,y_pred:np.ndarray, name:str):

    y_test = y_test.reshape(1,-1)

    y_pred = y_pred.reshape(1,-1)

    plt.figure(figsize=(9,7))

    plt.scatter(y_pred, (y_pred - y_test), c='orange', marker='*', s=63, edgecolors="black", label='Test data')

    plt.xlabel('Predicted values')

    plt.ylabel('Residuals')

    plt.title('Residual Analysis of '+ name )

    plt.legend(loc='upper right')

    plt.hlines(y=0, xmin=np.min(y_test)-1, xmax=np.max(y_test)+1, lw=2, color='red')

    plt.grid(True)

    plt.xlim([np.min(y_pred)-3, np.max(y_pred)+3])

    plt.show()



# To 10-Fold Cross calculate
```

```
def FoldCross(X:np.ndarray, Y:np.ndarray, draw:str):
    regname = ["Linear", "Ridge", "Lasso", "ElasticNet", "K-NN", "Decision Tree", "Random Forest", "SVR"]
    Kf_reg_list = []
    foldname = ["1. Fold","2. Fold","3. Fold","4. Fold","5. Fold","6. Fold","7. Fold","8. Fold","9. Fold","10. Fold"]
    regressionCode = ["l","R","L","E","K","D","RF","S"]
    skf = StratifiedKFold(n_splits=10, shuffle=True)
    for name in regressionCode:
        Temp = []
        i=1
        for train,test in skf.split(X,Y):
            R = Multi_RegRession(X[train],Y[train],X[test],Y[test],name)
            Temp.append([round(R[2],3),round(R[3],3)])
            if draw == "YES":
                reg_comparison(X[train],Y[train],X[test],Y[test], str(i)+". Fold")
                resudual_analysis(Y[test],R[1],name+" - "+str(i)+". Fold")
            i+=1
        Kf_reg_list.append(Temp)
    Kf = pd.DataFrame(Kf_reg_list, columns=foldname)
    df = []
    for i in range(1,11):
        temp = []
        for tup in Kf[str(i)+". Fold"]:
            temp.append(tup)
        df.append(temp)
```

```
    return pd.DataFrame(df, index=foldname, columns=regname)


# To compare the Training Algorithms
def reg_comparison(X_train:np.ndarray,y_train:np.ndarray,X_test:np.ndarray,y_test:np.ndarray,title:str):
    regressionCode = ["l","R","L","E","K","D","RF","S"]
    regname = ["Linear", "Ridge", "Lasso", "ElasticNet", "K-NN", "Decision Tree", "Random Forest", "SVR"]
    plt.figure(figsize=(13,7))
    sns.set_style("darkgrid")
    plt.plot(y_test, label="Real values", color="black")
    i=0
    for code in regressionCode:
        pred = Multi_RegRession(X_train,y_train,X_test,y_test,code)[1]
        plt.plot(pred, label=regname[i]+" (Predicts)")
        i+=1
    plt.title(title)
    plt.legend()
    plt.show()


# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\  QUESTİON _ 1 //////////////////////
# READING THE DATA FROM FILE
data = pd.read_csv("forestfires.csv")


#\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ QUESTİON _ 2 //////////////////////
# No-Preprocessing Data
data = pd.DataFrame(data, index=data.index, columns=data.columns)
```

```
X_columns = ["X","Y","month","day","DMC","DC","ISI","temp","RH","wind","rain","area"] #--> Independent variables

y_column = ["FFMC"] #--> Dependent variable (Target)

data_x = data[X_columns].values

data_y = data[y_column].values

data_y = data_y.astype('int')

sns.set_style('darkgrid')

sns.pairplot(data,x_vars=X_columns,y_vars=y_column,kind='reg',height=5, plot_kws={'color':'orange',

   'scatter_kws':{'color':'black'}})

plt.title("NO-PREPPROCESSİON DATA")

plt.show()



#  Standartized Data

scaler = StandardScaler().fit(data)

standartizedData = scaler.transform(data)

standartizedData = pd.DataFrame(standartizedData, index = data.index, columns=data.columns)

Sdata_x = standartizedData[X_columns].values

Sdata_y = standartizedData[y_column].values

Sdata_y = Sdata_y.astype('int')

sns.set_style('darkgrid')

sns.pairplot(standartizedData,x_vars=X_columns,y_vars=y_column,kind='reg',height=5, plot_kws={'color':'m',

   'scatter_kws':{'color':'black'}})

plt.title("STANDARTİZED DATA")

plt.show()



#\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ QUESTİON _ 3 ///////////////////////
```

```
# Dividing No-Prep. data into Learning and Testing

X_train, X_test, Y_train, Y_test = train_test_split(data_x,data_y,test_size=0.2, random_state=99)

reg_comparison(X_train,Y_train,X_test,Y_test,"Prediction of No-Prepocessing Data Regressions")



# Dividing No-Prep. data into Learning and Testing

X_train_S, X_test_S, Y_train_S, Y_test_S = train_test_split(Sdata_x,Sdata_y,test_size=0.2, random_state=99)

reg_comparison(X_train_S,Y_train_S,X_test_S,Y_test_S,"Prediction of Standartized Data Regressions")



#============================ NO-PREPROCESSING DATA ================================#

# Linear Regression of No-Preprocessing Data & R2 Score and MSE

reg = Multi_RegRession(X_train, Y_train,X_test,Y_test,"l")

DataLinearModel = reg[0]

dataLinear_Y_pred = reg[1]

dataLinearR2 = reg[2]

dataLinearMSE = reg[3]

print("R2 Score of Linear regression data: ",dataLinearR2)

print("MSE Score of Linear regression data: ",dataLinearMSE)

resudual_analysis(Y_test,dataLinear_Y_pred,"Linear reg. of no-prep. data")



# Ridge Regression of No-Preprocessing Data & R2 Score and MSE

reg = Multi_RegRession(X_train, Y_train,X_test,Y_test,"R")

DataRidgeModel = reg[0]

dataRidge_y_pred = reg[1]

dataRidgeR2 = reg[2]

dataRidgeMSE = reg[3]
```

```
print("Ridge R2 Score : ",dataRidgeR2)

print("Ridge MSE Score : ", dataRidgeMSE)

resudual_analysis(Y_test,dataRidge_y_pred,"Ridge reg. of no-prep. data")



# Lasso Regression of No-Preprocessing Data & R2 Score and MSE

reg = Multi_RegRession(X_train, Y_train,X_test,Y_test,"L")

DataLassoModel = reg[0]

dataLasso_y_pred = reg[1]

dataLassoR2 = reg[2]

dataLassoMSE = reg[3]

print("Lasso R2 Score : ",dataLassoR2)

print("Lasso MSE Score : ", dataLassoMSE)

resudual_analysis(Y_test,dataLasso_y_pred,"Lasso reg. of no-prep. data")



# Elastic_Net Regression of No-Preprocessing Data & R2 Score and MSE

reg = Multi_RegRession(X_train, Y_train,X_test,Y_test,"E")

DataElasticModel = reg[0]

dataElastic_y_pred = reg[1]

dataElasticR2 = reg[2]

dataElasticMSE = reg[3]

print("Elastic R2 Score : ",dataElasticR2)

print("Elastic MSE Score : ", dataElasticMSE)

resudual_analysis(Y_test,dataElastic_y_pred,"Elastic reg. of no-prep. data")



# KNN Regression of No-Preprocessing Data & R2 Score and MSE
```

```
reg = Multi_RegRession(X_train, Y_train,X_test,Y_test,"K")

DataKnnModel = reg[0]

dataKnn_y_pred = reg[1]

dataKnnR2 = reg[2]

dataKnnMSE = reg[3]

print("KNN R2 Score : ",dataKnnR2)

print("KNN MSE Score : ", dataKnnMSE)

resudual_analysis(Y_test,dataKnn_y_pred,"KNN reg. of no-prep. data")



# Decision Tree Regression of No-Preprocessing Data & R2 Score and MSE

reg = Multi_RegRession(X_train, Y_train,X_test,Y_test,"D")

DataTreeModel = reg[0]

dataTree_y_pred = reg[1]

dataTreeR2 = reg[2]

dataTreeMSE = reg[3]

print("Decision Tree R2 Score : ",dataTreeR2)

print("Decision Tree MSE Score : ", dataTreeMSE)

resudual_analysis(Y_test,dataTree_y_pred,"Decision reg. of no-prep. data")



# Random Forest Regression of No-Preprocessing Data & R2 Score and MSE

reg = Multi_RegRession(X_train, Y_train,X_test,Y_test,"RF")

DataRandomForestModel = reg[0]

dataRandomForest_y_pred =reg[1]

dataRandomForestR2 = reg[2]

dataRandomForestMSE = reg[3]
```

```
print("Random Forest R2 Score : ", dataRandomForestR2)

print("Random Forest MSE Score: ", dataRandomForestMSE)

resudual_analysis(Y_test,dataRandomForest_y_pred,"Random Forest reg. of no-prep. data")



# SVR  of No-Preprocessing Data & R2 Score and MSE

reg = Multi_RegRession(X_train, Y_train,X_test,Y_test,"S")

DataSVR_Model = reg[0]

dataSVR_y_pred = reg[1]

dataSVR_R2 = reg[2]

dataSVR_MSE = reg[3]

print("Support Vector Regression R2 Score : ",dataSVR_R2)

print("Support Vektor Regression MSE : ", dataSVR_MSE)

resudual_analysis(Y_test,dataSVR_y_pred,"SVR reg. of no-prep. data")



#================================== STANDARTIZED DATA ==========================================#

# Linear Regression of Standartized Data & R2 Score and MSE

reg = Multi_RegRession(X_train_S, Y_train_S, X_test_S,Y_test_S, "l")

SdataLinearModel = reg[0]

SdataLinear_y_pred = reg[1]

SdataLinear_R2 = reg[2]

SdataLinear_MSE = reg[3]

print("Linear  Regression R2 Score : ",SdataLinear_R2)

print("Linear Regression MSE : ", SdataLinear_MSE)

resudual_analysis(Y_test_S,SdataLinear_y_pred,"Linear reg. of Standartized data")
```

```
# Ridge Regression of Standartized Data & R2 Score and MSE

reg = Multi_RegRession(X_train_S, Y_train_S, X_test_S,Y_test_S, "R")

SdataRigeModel = reg[0]

SdataRidge_y_pred = reg[1]

SdataRidge_R2 = reg[2]

SdataRidge_MSE = reg[3]

print("Ridge  Regression R2 Score : ",SdataRidge_R2)

print("Ridge Regression MSE : ", SdataRidge_MSE)

resudual_analysis(Y_test_S,SdataRidge_y_pred,"Linear reg. of Standartized data")



# Lasso Regression of Standartized Data & R2 Score and MSE

reg = Multi_RegRession(X_train_S, Y_train_S, X_test_S,Y_test_S, "L")

SdataLassoModel = reg[0]

SdataLasso_y_pred = reg[1]

SdataLasso_R2 = reg[2]

SdataLasso_MSE = reg[3]

print("Lasso  Regression R2 Score : ",SdataLasso_R2)

print("Lasso Regression MSE : ", SdataLasso_MSE)

resudual_analysis(Y_test_S,SdataLasso_y_pred,"Lasso reg. of Standartized data")



# ElasticNet Regression of Standartized Data & R2 Score and MSE

reg = Multi_RegRession(X_train_S, Y_train_S, X_test_S,Y_test_S, "E")

SdataElasticModel = reg[0]

SdataElastic_y_pred = reg[1]

SdataElastic_R2 = reg[2]
```

```
SdataElastic_MSE = reg[3]

print("ElasticNet  Regression R2 Score : ",SdataRidge_R2)

print("ElasticNet Regression MSE : ", SdataRidge_MSE)

resudual_analysis(Y_test_S,SdataElastic_y_pred,"Elastic reg. of Standartized data")



# K-NN Regression of Standartized Data & R2 Score and MSE

reg = Multi_RegRession(X_train_S, Y_train_S, X_test_S,Y_test_S, "K")

SdataKnnModel = reg[0]

SdataKnn_y_pred = reg[1]

SdataKnn_R2 = reg[2]

SdataKnn_MSE = reg[3]

print("K-NN  Regression R2 Score : ",SdataKnn_R2)

print("K-NN Regression MSE : ", SdataKnn_MSE)

resudual_analysis(Y_test_S,SdataRidge_y_pred,"K-NN reg. of Standartized data")



# Decision Tree Regression of Standartized Data & R2 Score and MSE

reg = Multi_RegRession(X_train_S, Y_train_S, X_test_S,Y_test_S, "D")

SdataTreeModel = reg[0]

SdataTree_y_pred = reg[1]

SdataTree_R2 = reg[2]

SdataTree_MSE = reg[3]

print(" Decision Tree  Regression R2 Score : ",SdataTree_R2)

print("Decision Tree Regression MSE : ", SdataTree_MSE)

resudual_analysis(Y_test_S,SdataTree_y_pred,"Decision Tree reg. of Standartized data")
```

```
# Random Forest Regression of Standartized Data & R2 Score and MSE

reg = Multi_RegRession(X_train_S, Y_train_S, X_test_S,Y_test_S, "RF")

SdataRandomForestModel = reg[0]

SdataRandomForest_y_pred = reg[1]

SdataRandomForest_R2 = reg[2]

SdataRandomForest_MSE = reg[3]

print(" Random Forest  Regression R2 Score : ",SdataRandomForest_R2)

print("Random Forest Regression MSE : ", SdataRandomForest_MSE)

resudual_analysis(Y_test_S,SdataRandomForest_y_pred,"Random Forest reg. of Standartized data")



# Support Vector Regressor Regression of Standartized Data & R2 Score and MSE

reg = Multi_RegRession(X_train_S, Y_train_S, X_test_S,Y_test_S, "S")

SdataSVR_Model = reg[0]

SdataSVR_y_pred = reg[1]

SdataSVR_R2 = reg[2]

SdataSVR_MSE = reg[3]

print(" Support Vector Regressor  Regression R2 Score : ",SdataSVR_R2)

print("Support Vector Regressor Regression MSE : ", SdataSVR_MSE)

resudual_analysis(Y_test_S,SdataSVR_y_pred,"SVR reg. of Standartized data")



#\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ QUESTİON _ 4 //////////////////////

# No-Preprocession Data

FoldCross(data_x,data_y,"YES")



# Standartized Data
```

```
FoldCross(Sdata_x,Sdata_y,"YES")



#\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ QUESTİON _ 5 ///////////////////////

# ABSOLUTE CROSS - CORRELATİON  (Half of The Best Correlation) - No_preprocession Data

highest_corr = data.corr()["FFMC"].abs().nlargest(len(data.columns)//2)

print(highest_corr)

highest_data_X = pd.DataFrame(data[highest_corr.index],index=data.index, columns=highest_corr.index)

highest_data_X = highest_data_X[["ISI","temp","DMC","DC","RH"]].values

Score_N = FoldCross(highest_data_X,data_y,"YES")

print(Score_N)



# ABSOLUTE CROSS - CORRELATİON  (Half of The Best Correlation) - Standartized Data

highest_corr = standartizedData.corr()["FFMC"].abs().nlargest(len(data.columns)//2)

print(highest_corr)

highest_Sdata_X = pd.DataFrame(data[highest_corr.index],index=data.index, columns=highest_corr.index)

highest_Sdata_X = highest_Sdata_X[["ISI","temp","DMC","DC","RH"]].values

Score_S = FoldCross(highest_data_X,data_y,"YES")

print(Score_S)



#\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ QUESTİON _ 6 ///////////////////////

# No-Preprocession Data

X_columns = ["X","Y","month","day","DMC","DC","ISI","temp","RH","wind","rain","area"]

best_features = []

for column in range(0,12):

    d_x = pd.Series(data=data_x[:,column]).values.reshape(-1,1)
```

```
    X_train, X_test, Y_train, Y_test = train_test_split(d_x,data_y,test_size=0.2, random_state=99)

    Mse = Multi_RegRession(X_train,Y_train,X_test,Y_test,"I")[3]

    best_features.append([X_columns[column],Mse])

best_features = pd.DataFrame(data=best_features,
columns=["name","Score"]).sort_values(by=['Score']).nsmallest(6,columns=['Score'])

print(best_features) # The Best 6 Atribute Scores

Best_d_x = data[best_features['name']]

print(Best_d_x)

Result_N = FoldCross(Best_d_x.values,data_y,"YES")

print(Result_N)



# Standartized Data

X_columns = ["X","Y","month","day","DMC","DC","ISI","temp","RH","wind","rain","area"]

best_features = []

for column in range(0,12):

    Sd_x = pd.Series(data=Sdata_x[:,column]).values.reshape(-1,1)

    X_train_S, X_test_S, Y_train_S, Y_test_S = train_test_split(Sd_x,Sdata_y,test_size=0.2, random_state=99)

    Mse = Multi_RegRession(X_train_S,Y_train_S,X_test_S,Y_test_S,"I")[3]

    best_features.append([X_columns[column],Mse])

best_features = pd.DataFrame(data=best_features,
columns=["name","Score"]).sort_values(by=['Score']).nsmallest(6,columns=['Score'])

print(best_features) # The Best 6 Atribute Scores

Best_Sd_x = standartizedData[best_features['name']]

print(Best_Sd_x)

Result_S = FoldCross(Best_Sd_x.values,data_y,"YES")
```