DataAnalyse.py

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

data=pd.read_csv("cereal.csv")

##DATA HANDLING ===============================================================

data.drop('name',axis=1,inplace=True) # droped because unique data
#data.drop('type',axis=1,inplace=True) # droped because mostly same var.

##there are some negative valuse in data
data.replace(to_replace =-1,
                            value =np.NaN, inplace=True)

data.fillna(0, inplace=True)

#Label Encodincg to catogorical data*******************************************
from sklearn.preprocessing import LabelEncoder
# creating instance of labelencoder
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
data['mfr'] = labelencoder.fit_transform(data['mfr'])
data['type'] = labelencoder.fit_transform(data['type'])

data.drop('type',axis=1,inplace=True)


#Label Encodincg catogorical data END *****************************************

#FEATURE SELECTION START-------------------------------------------------------

x = data.drop("rating",1)
y = data["rating"]

#heatmap
plt.figure(figsize=(16,13))
cor = data.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()

##DATA HANDLING END ===========================================================
data.to_csv('cereal_clean.csv',index=False)


#feature importances grafical show----------------
plt.figure(figsize=(16, 9))

ranking = rf.feature_importances_
features = np.argsort(ranking)[::-1][:16]
columns = X.columns

plt.title("Feature importances based on Random Forest Regressor", y = 1.03, size = 18)
plt.bar(range(len(features)), ranking[features], color="aqua", align="center")
plt.xticks(range(len(features)), columns[features], rotation=80)
plt.show()
```

LineerRegression.py

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import datasets, linear_model
import pandas as pd
import numpy as np


import matplotlib
import matplotlib.pyplot as plt


# veri yukleme
data = pd.read_csv('cereal_clean.csv')

x = data.iloc[:,0:-1]
y = data.iloc[:,-1:]
X = x.values
Y = y.values


X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.33, random_state=0)


lm = linear_model.LinearRegression()
model = lm.fit(X_train, y_train)
predictions = lm.predict(X_test)
print( "Score: ", model.score(X_test, y_test))
```

```
plt.scatter(y_test, predictions)
plt.xlabel("True Values",size=10)
plt.ylabel("Predictions")

#R2 analiz
from sklearn.metrics import r2_score
print( "r2 Score: ")
print (r2_score(y_test,predictions))

#MSE analiz
from sklearn.metrics import mean_squared_error
print( "mse Score: ")
print (mean_squared_error(y_test,predictions))
```

Svr.py

```
#1. kutuphaneler
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# veri yukleme
data = pd.read_csv('cereal_clean.csv')

x = data.iloc[:,0:-1]
y = data.iloc[:,-1:]
X = x.values
Y = y.values


#verilerin olceklenmesi
from sklearn.preprocessing import StandardScaler

sc1 = StandardScaler()
x_olcekli = sc1.fit_transform(X)
sc2 = StandardScaler()
y_olcekli = sc2.fit_transform(Y)

X_train, X_test, y_train, y_test = train_test_split(x_olcekli, y_olcekli, test_size=0.2, random_state=25)

from sklearn.svm import SVR

svr_reg = SVR(kernel = 'linear')
svr_reg.fit(X_train,y_train)
```

```
predictions =svr_reg.predict(X_test)

plt.scatter(y_test,predictions,color='red')
#plt.plot(y_test,svr_reg.predict(X_test),color='blue')


#R2 analiz

from sklearn.metrics import r2_score
print( "r2 Score: ")
print (r2_score(y_test,predictions))

#MSE analiz
from sklearn.metrics import mean_squared_error
print( "mse Score: ")
print (mean_squared_error(y_test,predictions))
```

RandomForestsRegression.py

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split

from matplotlib import pyplot as plt



data=pd.read_csv("cereal_clean.csv")

#Random forest regression
X=data.drop('rating',axis=1)
y=data.rating

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

from sklearn.ensemble import RandomForestRegressor

cereal_model = RandomForestRegressor(n_estimators=15, random_state=30)#30
cereal_model.fit(X_train, y_train)


predictions = cereal_model.predict(X_test)

plt.scatter(y_test, predictions)
plt.xlabel("True Values")
plt.ylabel("Predictions")
#R2 analiz

from sklearn.metrics import r2_score
print( "r2 Score: ")
print (r2_score(y_test,predictions))

#MSE analiz
from sklearn.metrics import mean_squared_error
print( "mse Score: ")
print (mean_squared_error(y_test,predictions))

#MAE analiz
from sklearn.metrics import mean_absolute_error
print( "msa Score: ")
print (mean_absolute_error(y_test,predictions))
```

DecisionTrees.py

```
import pandas as pd
import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split


from matplotlib import pyplot as plt



data=pd.read_csv("cereal_clean.csv")

#Division of data into features and quality
X = data.iloc[:, 0:13].values
y = data.iloc[:, 13:14].values


x_train, x_test, y_train, y_test = train_test_split(X, y ,test_size=0.2,random_state=42)


sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.fit_transform(x_test)
Y_train = sc.fit_transform(y_train)
```

```
Y_test=sc.fit_transform(y_test)


from sklearn.tree import DecisionTreeRegressor

regr=DecisionTreeRegressor(random_state=0)#29
regr.fit(x_train,y_train)

predictions =regr.predict(x_test)

print("Decision Tree Testing score:",regr.score(x_test,y_test))
plt.scatter(y_test,predictions ,color='red')
plt.xlabel("True Values")
plt.ylabel("Predictions")

from sklearn.tree import export_graphviz

# export the decision tree to a tree.dot file
# for visualizing the plot easily anywhere

export_graphviz(regr, out_file ='treelimited.dot',
               feature_names =['mfr','calories','protein','fat','sodium','fiber','carbo','sugars','potass'
                class_names = 'rating',
                rounded = True, proportion = False, precision = 2, filled = True)


#R2 analiz
from sklearn.metrics import r2_score
print( "r2 Score: ")
print (r2_score(y_test,predictions))

#MSE analiz
from sklearn.metrics import mean_squared_error
print( "mse Score: ")
print (mean_squared_error(y_test,predictions))
```