Appendix

```
In [ ]: import pandas as pd
        import numpy as np
        import seaborn as  sns
        import matplotlib.pyplot as plot
        import warnings
        warnings.filterwarnings('ignore')
```

```
In [ ]: # Laod the data set
        df = pd.read_csv("C:\project\concrete_data.csv")
```

```
In [ ]: df
```

```
In [ ]: df.head()
```

```
In [ ]: df.info()
```

```
In [ ]: df.columns
```

```
In [ ]: df.shape
```

```
In [ ]: df.describe().T
```

```
In [ ]: df.dtypes
```

```
In [ ]: # To check the null values
        df.isnull().sum()
```

```
In [ ]: sns.heatmap(df.isnull(), yticklabels=False, annot= True)
```

EDA

```
In [ ]: corelation = df.corr()
```

```
In [ ]: import matplotlib.pyplot as plt
```

```
In [ ]: plt.figure(figsize = (15,10))
        sns.heatmap(corelation,xticklabels=corelation.columns,yticklabels=corelation.columns,annot=True, cmap="viridis"
        plt.title("corelation between different variables")
```

```
In [ ]: # Pairplot
        sns.pairplot(df,diag_kind="kde")
```

```
In [ ]: fig,ax =plt.subplots(figsize = (7,6))
        sns.scatterplot(y= "concrete_compressive_strength",x = "cement",hue="water", size = "age",data=df,ax=ax,sizes=(
        ax.set_title("concrete_compressive_strength vs (cement, age, water)")
        ax.legend(loc="upper left", bbox_to_anchor = (1,1))
        plt.show()
```

```
In [ ]: fig,ax =plt.subplots(figsize = (7,6))
        sns.scatterplot(y= "concrete_compressive_strength",x = "fine_aggregate ",hue="fly_ash", size = "superplasticize
        ax.set_title("concrete_compressive_strength vs (fine_aggregate , fly_ash, superplasticizer)")
        ax.legend(loc="upper left", bbox_to_anchor = (1,1))
        plt.show()
```

Checking the outliers

```
In [ ]: from scipy import stats

        Q1 = df['cement'].quantile(q=0.25)
        Q3 = df['cement'].quantile(q=0.75)

        print('1st Quartile (Q1) is: ', Q1)
        print('3rd Quartile (Q3) is: ', Q3)
        print('Interquartile range (IQR) is ', stats.iqr(df['cement']))
```

```
In [ ]: L_outliers = Q1-1.5*(Q3-Q1)
        U_outliers = Q3+1.5*(Q3-Q1)
        print('Lower outlier limit in cement: ', L_outliers)
        print('Upper outlier limit in cement: ', U_outliers)
```

```
In [ ]: print('Number of outliers in cement upper: ', df[df['cement'] > 586.4375]['cement'].count())
```

```
print('Number of outliers in cement lower: ', df[df['cement'] < -44.0625]['cement'].count())
```

In [ ]:
```
# Boxplot
sns.boxplot(x='cement', data=df, orient='h')
```

In [ ]:
```
#Water
w_Q1 = df['water'].quantile(q=0.25)
w_Q3 = df['water'].quantile(q=0.75)

print('1st Quartile (Q1) is: ', w_Q1)
print('3rd Quartile (Q3) is: ', w_Q3)
print('Interquartile range (IQR) is: ', stats.iqr(df['water']))
```

In [ ]:
```
WL_outliers = w_Q1 - 1.5*(w_Q3-w_Q1)
WU_outliers = w_Q3 + 1.5*(w_Q3-w_Q1)

print('Lower outlier in water: ',WL_outliers)
print('Upper outlier in water: ',WU_outliers)
```

Distplot

In [ ]:
```
#Boxplot
sns.boxplot(x='water', data=df, orient='h')
```

In [ ]:
```
Q1 = df['blast_furnace_slag'].quantile(q=0.25)
Q3 = df['blast_furnace_slag'].quantile(q=0.75)
```

In [ ]:
```
L_outliers = Q1 - 1.5*(Q3-Q1)
U_outliers = Q3 + 1.5*(Q3-Q1)

print('Lower outlier in water: ', L_outliers)
print('Upper outlier in water: ', U_outliers)
```

In [ ]:
```
print('Number of outliers in slag upper: ', df[df['blast_furnace_slag'] > 357.375]['blast_furnace_slag'].count(
print('Number of outliers in slag lower: ', df[df['blast_furnace_slag'] < -214.425]['blast_furnace_slag'].count
```

In [ ]:
```
# Boxplot
sns.boxplot(x='blast_furnace_slag', data=df, orient='h')
```

In [ ]:
```
Q1 = df['age'].quantile(q=0.25)
Q3 = df['age'].quantile(q=0.75)
```

In [ ]:
```
L_outliers = Q1 - 1.5*(Q3-Q1)
U_outliers = Q3 + 1.5*(Q3-Q1)

print('Lower outlier in age: ',L_outliers)
print('Upper outlier in age: ',U_outliers)
```

In [ ]:
```
print('Number of outliers in age upper: ', df[df['age'] > 129.5]['age'].count())
print('Number of outliers in age lower: ', df[df['age'] < -66.5]['age'].count())
```

In [ ]:
```
#Boxplot of age

sns.boxplot(x='age', data=df, orient='h')
```

In [ ]:
```
Q1 = df['fly_ash'].quantile(q=0.25)
Q3 = df['fly_ash'].quantile(q=0.75)
```

In [ ]:
```
L_outliers = Q1 - 1.5*(Q3-Q1)
U_outliers = Q3 + 1.5*(Q3-Q1)

print('Lower outlier in fly_ash: ', L_outliers)
print('Upper outlier in fly_ash: ', U_outliers)
```

In [ ]:
```
print('Number of outliers in fly_ash upper: ', df[df['fly_ash'] > 295.75]['fly_ash'].count())
print('Number of outliers in fly_ash lower: ', df[df['fly_ash'] < -177.45]['fly_ash'].count())
```

In [ ]:
```
# Boxplot of fly_ash
sns.boxplot(x='fly_ash', data=df, orient='h')
```

In [ ]:
```
df.boxplot(figsize=(15,9))
```

In [ ]:
```
print('Outliers in cement: ', df[((df.cement - df.cement.mean())/df.cement.std()).abs()>3]['cement'].count())
print('Outliers in blast_furnace_slag: ', df[((df.blast_furnace_slag - df.blast_furnace_slag.mean())/df.blast_f
print('Outliers in fly_ash: ', df[((df.fly_ash - df.fly_ash.mean())/df.fly_ash.std()).abs()>3]['fly_ash'].count
print('Outliers in water: ', df[((df.water - df.water.mean())/df.water.std()).abs()>3]['water'].count())
print('Outliers in superplasticizer: ', df[((df.superplasticizer - df.superplasticizer.mean())/df.superplastici
print('Outliers in coarse_aggregate: ', df[((df.coarse_aggregate - df.coarse_aggregate.mean())/df.coarse_aggreg
print('Outliers in age: ', df[((df.age - df.age.mean())/df.age.std()).abs()>3]['age'].count())
```

Replacing the outlier by median

In [ ]:
```
for cols in df.columns[:-1]:
  Q1 = df[cols].quantile(0.25)
  Q3 = df[cols].quantile(0.75)
  iqr = Q3 - Q1

  low = Q1-1.5*iqr
  high = Q3+1.5*iqr
  df.loc[(df[cols] < low) | (df[cols] > high), cols] = df[cols].median()
```

In [ ]:
```
df.boxplot(figsize=(15,9))
```

Distribution plot

In [ ]:
```
from matplotlib import colors
```

In [ ]:
```
plot.figure(figsize= (18,14),facecolor="white")
plotnumber = 1

for column in df.columns:
    ax = plot.subplot(4,3,plotnumber,)
    sns.distplot(df[column],color="red")
    plot.xlabel(column,fontsize = 10)
    plotnumber+=1
plot.show()
```

Standardize the data

In [ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [ ]:
```
x = df.drop(columns=["concrete_compressive_strength"])
```

In [ ]:
```
y = df.concrete_compressive_strength
```

In [ ]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.20,random_state=3)
```

In [ ]:
```
scalar = StandardScaler()
```

In [ ]:
```
scalar.fit(x_train)
```

In [ ]:
```
scalar.fit(x_test)
```

In [ ]:
```
x_train_std = scalar.transform(x_train)
x_test_std = scalar.transform(x_test)
```

In [ ]:
```
x_train1 = pd.DataFrame(x_train_std,columns=["cement","slag","fly_ash","water","superplasticizer","coarse_agg",
```

In [ ]:
```
x_test1 = pd.DataFrame(x_test_std,columns=["cement","slag","fly_ash","water","superplasticizer","coarse_agg","f
```

In [ ]:
```
plot.figure(figsize= (15,15), facecolor='white')
plotnumber = 1

for column in x_train.columns:
    ax = plot.subplot(4,3,plotnumber)
    sns.distplot(x_train[column])
    plot.xlabel(column,fontsize = 10)
    plotnumber+=1
plot.show()
```

Linear Regression

In [ ]:
```
from sklearn.linear_model import LinearRegression,Ridge,Lasso
```

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
lr=LinearRegression()
fit=lr.fit(x_train1,y_train)
score = lr.score(x_test1,y_test)


print('..................................')
```

```
y_predict = lr.predict(x_test1)
print('mean_sqrd_error is ==',mean_squared_error(y_test,y_predict))
lr_rmse = np.sqrt(mean_squared_error(y_test,y_predict))
print('root mean squared error is == {}'.format(lr_rmse))
```

```
lr_score= print('score of Linear regression is:-',lr.score(x_test1,y_test))
```

```
Accuracy = print(lr.score(x_test1,y_test)*100,"%")
```

```
plt.figure(figsize=[12,8])
plt.scatter(y_predict,y_test)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red')
plt.xlabel('predicted')
plt.ylabel('orignal')
plt.title("Linear Regression")
plt.show()
```

```
rd= Ridge(alpha=0.4)
ls= Lasso(alpha=0.3)
fit_rd=rd.fit(x_train1,y_train)
fit_ls = ls.fit(x_train1,y_train)
print('score od ridge regression is:-',rd.score(x_test1,y_test))
print('.......................................................')
print('score of lasso is:-',ls.score(x_test1,y_test))
print('mean_sqrd_roor of ridig is==',mean_squared_error(y_test,rd.predict(x_test1)))
print('mean_sqrd_roor of lasso is==',mean_squared_error(y_test,ls.predict(x_test1)))
ridge_rmse = print('root_mean_squared error of ridge is==',np.sqrt(mean_squared_error(y_test,rd.predict(x_test1
lasso_rmse = print('root_mean_squared error of lasso is==',np.sqrt(mean_squared_error(y_test,lr.predict(x_test1
```

```
plt.figure(figsize=[12,8])
plt.scatter(y_predict,y_test)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red')
plt.xlabel('predicted')
plt.ylabel('orignal')
plt.title("Ridge Regression")
plt.show()
```

```
plt.figure(figsize=[12,8])
plt.scatter(y_predict,y_test)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red')
plt.xlabel('predicted')
plt.ylabel('orignal')
plt.title("Lasso Regression")

plt.show()
```

```
pd.DataFrame({"Algorithm":["Linear Regression","Lasso Regression"], "MSE":["91.92","93.31"],"RMSE":["9.58","9.5
```

Decision Tree

```
df.head()
```

```
x1 = df.iloc[:,:-1]
y1 = df.iloc[:,-1]
```

```
x1_train,x1_test,y1_train,y1_test = train_test_split(x1,y1,test_size=0.2,random_state=2)
```

```
from sklearn.tree import DecisionTreeRegressor
```

```
dt = DecisionTreeRegressor(max_depth=4)
```

```
dt.fit(x1_train , y1_train)
```

```
features = x1_train.columns
```

```
from sklearn import tree
plt.figure(figsize=(13,7))
tree.plot_tree(dt,filled=True)
```

```
Importance = pd.DataFrame({"Importance":dt.feature_importances_*100}, index=x.columns)
Importance.sort_values(by='Importance', axis=0, ascending=True).iloc[-10:,:].plot(kind='barh', color='green')
plt.xlabel("Variable Severity Levels")
```

```
from sklearn.metrics import r2_score
```

```
y1_pred_dt = dt.predict(x1_test)
```

```
print('mean_sqrd_error is ==',mean_squared_error(y1_test,y1_pred_dt))
```

```
dt_rmse = np.sqrt(mean_squared_error(y1_test,y1_pred_dt))
print('root mean squared error is == {}'.format(dt_rmse))
```

In [ ]:
```
dt_r2 =r2_score(y1_test,y1_pred_dt)
dt_r2
```

In [ ]:
```
dt.feature_importances_
```

In [ ]:
```
importances = pd.Series(dt.feature_importances_ , index=x1.columns)
importances.plot(kind = 'barh', figsize=(12,8))
```

from the decision tree model we can observe that top 4 contributors to strength are cement,age,water,superplasticizer,blast furnance slg

In [ ]:
```
pd.DataFrame({"Algorithm":["Decision Tree"], "MSE":"77.55","RMSE":"8.80","R^2":"0.68"})
```

Random Forest

In [ ]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [ ]:
```
rf = RandomForestRegressor()
```

In [ ]:
```
rf.fit(x1_train,y1_train)
```

In [ ]:
```
y1_pred_rf = rf.predict(x1_test)
```

In [ ]:
```
rf_r2 =r2_score(y1_test,y1_pred_rf)
rf_r2
```

In [ ]:
```
print('mean_sqrd_error is ==',mean_squared_error(y1_test,y1_pred_rf))
rf_rmse = np.sqrt(mean_squared_error(y1_test,y1_pred_rf))
print('root mean squared error is == {}'.format(rf_rmse))
```

In [ ]:
```
pd.DataFrame({"Algorithm":["Random Forest Regressor"], "MSE":"27.66","RMSE":"5.26","R^2":"0.88"})
```

In [ ]:
```
x1_predict = list(rf.predict(x1_test))
predicted_df = {'predicted_values': x1_predict, 'original_values': y1_test}


df2 = pd.DataFrame(predicted_df).head(20)
```

In [ ]:
```
df2
```

In [ ]:
```
df2.to_excel("my_file.xlsx")
```

In [ ]:
```
sns.regplot(x = "predicted_values", y = "original_values", data = df2)
```

In [ ]:
```
Importance = pd.DataFrame({"Importance":rf.feature_importances_*100}, index=x.columns)
Importance.sort_values(by='Importance', axis=0, ascending=True).iloc[-10:,:].plot(kind='barh', color='turquoise
plt.xlabel("Variable Severity Levels")
```

Support Vector Regressor

In [ ]:
```
from sklearn.svm import SVR
```

In [ ]:
```
svr = SVR(kernel="linear").fit(x1_train, y1_train)
```

In [ ]:
```
y1_pred_svr = svr.predict(x1_test)
```

In [ ]:
```
svr_r2 =r2_score(y1_test,y1_pred_svr)
svr_r2
```

In [ ]:
```
print('mean_sqrd_error is ==',mean_squared_error(y1_test,y1_pred_svr))
svr_rmse = np.sqrt(mean_squared_error(y1_test,y1_pred_svr))
print('root mean squared error is == {}'.format(svr_rmse))
```

In [ ]:
```
pd.DataFrame({"Algorithm":["Support Vector Regressor"], "MSE":"79.80","RMSE":"8.93","R^2":"0.67"})
```

K Nearest Neighbour

In [ ]:
```
from sklearn.neighbors import KNeighborsRegressor
```

In [ ]:
```
knn = KNeighborsRegressor()
```

In [ ]:
```
knn.fit(x1_train,y1_train)
```

In [ ]:
```
y1_pred_knn = knn.predict(x1_test)
```

In [ ]:
```
knn_r2 = r2_score(y1_test,y1_pred_knn)
knn_r2
```

In [ ]:
```
print('mean_sqrd_error is ==',mean_squared_error(y1_test,y1_pred_knn))
knn_rmse = np.sqrt(mean_squared_error(y1_test,y1_pred_knn))
print('root mean squared error is == {}'.format(knn_rmse))
```

In [ ]:
```

```

In [ ]:
```
pd.DataFrame({"Algorithm":["K Nearest Neighbour"], "MSE":"86.61","RMSE":"9.30","R^2":"0.64"})
```

Ada Boost Regressor

In [ ]:
```
from sklearn.ensemble import AdaBoostRegressor
```

In [ ]:
```
adb =AdaBoostRegressor()
```

In [ ]:
```
adb.fit(x1_train,y1_train)
```

In [ ]:
```
y1_pred_adb = adb.predict(x1_test)
```

In [ ]:
```
adb_r2= r2_score(y1_test,y1_pred_adb)
adb_r2
```

In [ ]:
```
print('mean_sqrd_error is ==',mean_squared_error(y1_test,y1_pred_adb))
adb_rmse = np.sqrt(mean_squared_error(y1_test,y1_pred_adb))
print('root mean squared error is == {}'.format(adb_rmse))
```

In [ ]:
```
pd.DataFrame({"Algorithm":["Ada Boost Regressor"], "MSE":"67.97","RMSE":"8.24","R^2":"0.72"})
```

In [ ]:
```
df3=pd.DataFrame({"Algorithm":["Linear Regression",
                           "Lasso Regression",

                           "Decision Tree",
                           "Random Forest",
                           "Support Vector Regression",
                           "K Nearest Neighbour",
                           "Ada Boost regressor"],
                   "R_sq":["0.68","0.68","0.75","0.88","0.67","0.64","0.72"],
                  "RMSE":["9.58","9.58","7.80","5.26","8.93","9.30","8.24"]})
```

In [ ]:
```
df3
```

In [ ]:
```
df3.to_excel("my_file2.xlsx")
```

In [ ]:
```
sns.scatterplot(x="R_sq",y="RMSE",data=df3,hue="Algorithm")
```

In [ ]:
```
df3.info()
```

In [ ]:
```
df3.RMSE=df3.RMSE.astype("float")
```

In [ ]:
```
df3.R_sq=df3.R_sq.astype("float")
```

In [ ]:
```
names = ["Linear Regression", "Lasso Regression", "Dcision Tree Regressor", "Random Forest Regressor","Support
fig = plt.figure(figsize=(20,12))

df3[["Algorithm","RMSE"]].plot(kind="bar",figsize=(15,8))
plt.ylabel('RMSE')
plt.xticks([0,1,2,3,4,5,6],names,rotation=45)
plt.xlabel('Models')
plt.title('RMSE with Different Algorithms')
# plt.xticks(names, rotation=45)
```

In [ ]:
```
import matplotlib.cm as cm
```

In [ ]:
```
names = ["Linear Regression", "Lasso Regression", "Dcision Tree Regressor", "Random Forest Regressor","Support
fig = plt.figure(figsize=(20,12))

df3[["Algorithm","R_sq"]].plot(kind="bar",figsize=(15,8))
plt.ylabel('R_sq')
```

```python
plt.xticks([0,1,2,3,4,5,6],names,rotation=45)
plt.xlabel('Models')
plt.title('R_sq with Different Algorithms')
# plt.xticks(names, rotation=45)
```