# Course Work 1 – Code

# -\*- coding: utf-8 -\*-

"""Untitled2.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1X-gyZkIFVKA9ZWEOEjzqBt4bWultPc1o

"""

import pandas as pd

import numpy as np

df=pd.read\_csv("/content/diamonds\_coursework.csv", na\_values='??')

df.info

df.drop('Unnamed: 11', inplace=True, axis=1)

df.drop('Unnamed: 12', inplace=True, axis=1)

df.info()

df.dropna(inplace=True)

df.info()

df.drop('id', inplace=True, axis=1)

df.info()

# Commented out IPython magic to ensure Python compatibility.

import sklearn

assert sklearn.\_\_version\_\_ >= "0.20"

import os

# %matplotlib inline

import matplotlib as mpl

import matplotlib.pyplot as plt

mpl.rc('axes', labelsize=14)

mpl.rc('xtick', labelsize=12)

mpl.rc('ytick', labelsize=12)

df.info()

df["price"].value\_counts()

df.describe()

# Commented out IPython magic to ensure Python compatibility.

# %matplotlib inline

import matplotlib.pyplot as plt

df.hist(bins=50, figsize=(20,15))

plt.show()

log\_binsize = 0.025

bins = 10 \*\* np.arange(2.4, np.log10(df['price'].max())+log\_binsize, log\_binsize)

plt.figure(figsize=[8, 5])

plt.hist(data = df, x = 'price', bins = bins)

plt.xscale('log')

plt.xticks([500, 1e3, 2e3, 5e3, 1e4, 2e4], [500, '1k', '2k', '5k', '10k', '20k'])

plt.xlabel('Price ($)')

plt.show()

import seaborn as sns

df= sns.load\_dataset("diamonds")

sns.catplot(data=df, x="cut", kind="count")

import seaborn as sns

df= sns.load\_dataset('diamonds')

sns.catplot(data=df, kind="bar", x="color", y="price", aspect=1.5)

sns.catplot(data=df, x="carat", y="price", kind='bar')

sns.catplot(data=df, x="cut", y="price", kind='bar')

sns.catplot(data=df, x='clarity', y='price', kind='bar')

import seaborn as sns

df= sns.load\_dataset("diamonds")

sns.catplot(data=df, x="color", kind="count")

import matplotlib.pyplot as plt

labels = df.clarity.unique().tolist()

sizes = df.clarity.value\_counts().tolist()

colors = ['#006426', '#e44800', '#a0097f', '#614d05', '#fff70d', '#16f5b2','#ff9999','#66b3ff']

explode = (0.1, 0.0, 0.1, 0, 0.1, 0, 0.1,0)

plt.pie(sizes, explode=explode, labels=labels, colors=colors,autopct='%1.1f%%', shadow=True, startangle=0)

plt.axis('equal')

plt.title("Percent of Clarity Categories")

plt.plot()

fig=plt.gcf()

fig.set\_size\_inches(6,6)

plt.show()

plt.figure(figsize=(5,5))

plt.hist( x=df['carat'] , bins=30 ,color='b')

plt.xlabel('carat')

plt.ylabel('Frequency')

plt.title('Carat')

plt.xlim(0,3)

plt.ylim(0,12000)

#plotting carat #gajar

binsize = 0.01

bins = np.arange(0.2, 1.51, 0.01)

plt.figure(figsize=[8, 5])

plt.hist(data = df, x = 'carat', bins = bins)

plt.xlim([0.2,1.5])

plt.xlabel('Carat')

plt.show()

#plotting cut,clarity,colour

fig, ax = plt.subplots(nrows=3, figsize = [8,8])

default\_color = sns.color\_palette()[0]

sns.countplot(data = df, x = 'cut', color = default\_color, ax = ax[0])

sns.countplot(data = df, x = 'color', color = default\_color, ax = ax[1])

sns.countplot(data = df, x = 'clarity', color = default\_color, ax = ax[2])

plt.show()

np.random.seed(42)

corr = df.corr()

fig = plt.figure()

ax = fig.add\_subplot(111)

cax = ax.matshow(corr,cmap='coolwarm', vmin=-1, vmax=1)

fig.colorbar(cax)

ticks = np.arange(0,len(df.columns),1)

ax.set\_xticks(ticks)

plt.xticks(rotation=90)

ax.set\_yticks(ticks)

ax.set\_xticklabels(df.columns)

ax.set\_yticklabels(df.columns)

plt.show()

corr

import seaborn as sns

import matplotlib.pyplot as plt

from pandas.plotting import scatter\_matrix

attributes = ['price','cut','color', 'clarity', 'carat', 'table', 'depth']

scatter\_matrix(df[attributes], figsize = (12,8))

from sklearn import datasets,linear\_model

from sklearn.model\_selection import train\_test\_split

def split\_train\_test(data, test\_ratio):

shuffled\_indices = np.random.permutation(len(data))

test\_set\_size = int(len(data) \* test\_ratio)

test\_indices = shuffled\_indices[:test\_set\_size]

train\_indices = shuffled\_indices[test\_set\_size:]

return data.iloc[train\_indices], data.iloc[test\_indices]

train\_set, test\_set = train\_test\_split(df, test\_size = 0.3, random\_state = 42)

print(f"Rows in train set: {len(train\_set)}\nRows in test set: {len(test\_set)}\n")

test\_set.head()

train\_set.head()

test\_set=pd.get\_dummies(data=test\_set, drop\_first=True)

test\_set.head()

train\_set=pd.get\_dummies(data=train\_set, drop\_first=True)

train\_set.head()

from sklearn import linear\_model

X= train\_set[['carat','table','cut\_Good','cut\_Ideal','cut\_Premium','color\_H','color\_I','color\_J', 'clarity\_IF','clarity\_SI1', 'clarity\_SI2' ,'clarity\_VS1','clarity\_VS2','clarity\_VVS1','clarity\_VVS2']]

y= train\_set['price']

regr = linear\_model.LinearRegression()

regr.fit(X, y)

print(regr.coef\_)

from sklearn.metrics import mean\_squared\_error

df\_predictions = regr.predict(X)

df\_predictions

mse= mean\_squared\_error(y,df\_predictions)

rmse=np.sqrt(mse)

rmse

from sklearn.metrics import mean\_absolute\_error

mae=mean\_absolute\_error(y,df\_predictions)

mae

from sklearn.metrics import r2\_score

r2=r2\_score(y,df\_predictions)

r2

print('Linear Regression model')

print('RMSE Values:', rmse)

print('MAE Value:', mae)

print('R Squared:', r2)

from sklearn.ensemble import RandomForestRegressor

forest = RandomForestRegressor(n\_estimators=100, random\_state=42)

forest.fit(X,y)

df\_pred=forest.predict(X)

forest\_mse = mean\_squared\_error(y,df\_pred)

forest\_rmse = np.sqrt(forest\_mse)

forest\_rmse

forest\_mae=mean\_absolute\_error(y,df\_pred)

forest\_mae

from sklearn.metrics import r2\_score

forest\_r2=r2\_score(y,df\_pred)

forest\_r2

from sklearn.svm import SVR

svm\_reg = SVR(kernel="linear")

svm\_reg.fit(X, y)

df\_pred2 = svm\_reg.predict(X)

svm\_mse = mean\_squared\_error(y, df\_pred2)

svm\_rmse = np.sqrt(svm\_mse)

svm\_rmse

svm\_mae=mean\_absolute\_error(y,df\_pred2)

svm\_mae

from sklearn.metrics import r2\_score

svm\_r2=r2\_score(y,df\_pred2)

svm\_r2

from sklearn.model\_selection import cross\_val\_score

scores = cross\_val\_score(regr,X,y, scoring='neg\_mean\_squared\_error', cv=10)

regr\_score=np.sqrt(-scores)

regr\_score

def display\_scores(scores):

print("Scores:", scores)

print("Mean:", scores.mean())

print("Standard deviation:", scores.std())

print('Cross validation scores for Linear regression')

display\_scores(regr\_score)

scores = cross\_val\_score(forest,X,y, scoring='neg\_mean\_squared\_error', cv=10)

forest\_score=np.sqrt(-scores)

forest\_score

print('Cross validation scores for Random forest regression')

display\_scores(forest\_score)

from sklearn.model\_selection import GridSearchCV

param\_grid = [

# try 12 (3×4) combinations of hyperparameters

{'n\_estimators': [3, 10, 30], 'max\_features': [2, 4, 6, 8]},

# then try 6 (2×3) combinations with bootstrap set as False

{'bootstrap': [False], 'n\_estimators': [3, 10], 'max\_features': [2, 3, 4]},

]

forest\_reg = RandomForestRegressor(random\_state=42)

# train across 5 folds, that's a total of (12+6)\*5=90 rounds of training

grid\_search = GridSearchCV(forest\_reg, param\_grid, cv=5,

scoring='neg\_mean\_squared\_error',

return\_train\_score=True)

grid\_search.fit(X,y)

grid\_search.best\_params\_

cvres = grid\_search.cv\_results\_

for mean\_score, params in zip(cvres["mean\_test\_score"], cvres["params"]):

print(np.sqrt(-mean\_score), params)

pd.DataFrame(grid\_search.cv\_results\_)

import numpy as np

from sklearn import datasets

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import learning\_curve

import matplotlib.pyplot as plt

train\_size,train\_score,validation\_score = learning\_curve(regr,X,y,train\_sizes=np.logspace(-1,0,3))

plt.plot(train\_size,validation\_score.mean(axis=1),lw=2,label='cross-validation')

plt.plot(train\_size,train\_score.mean(axis=1),lw=2, label='training')

plt.xlabel('number of data set')

plt.ylabel('Accuracy score')

plt.title('Learning Curve:Linear Regression')

plt.legend(['cross validation', 'training score'], loc='best')

train\_size,train\_score,validation\_score = learning\_curve(forest,X,y,train\_sizes=np.logspace(-1,0,3))

plt.plot(train\_size,validation\_score.mean(axis=1),lw=2,label='cross-validation')

plt.plot(train\_size,train\_score.mean(axis=1),lw=2, label='training')

plt.xlabel('number of data set')

plt.ylabel('Accuracy score')

plt.title('Learning Curve:Random forest Regression')

plt.legend(['cross validation', 'training score'], loc='best')

train\_size,train\_score,validation\_score = learning\_curve(svm\_reg,X,y,train\_sizes=np.logspace(-1,0,3))

plt.plot(train\_size,validation\_score.mean(axis=1),lw=2,label='cross-validation')

plt.plot(train\_size,train\_score.mean(axis=1),lw=2, label='training')

plt.xlabel('number of data set')

plt.ylabel('Accuracy score')

plt.title('Learning Curve: Support Vector Machine')

plt.legend(['cross validation', 'training score'], loc='best')