The environment didnt let me download the assignment in any way, so here it is in a raw form:

# Surpress warnings:

def warn(\*args, \*\*kwargs):

pass

import warnings

warnings.warn = warn

import piplite

await piplite.install('seaborn')

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler,PolynomialFeatures

from sklearn.linear\_model import LinearRegression

%matplotlib inline

from pyodide.http import pyfetch

async def download(url, filename):

response = await pyfetch(url)

if response.status == 200:

with open(filename, "wb") as f:

f.write(await response.bytes())

filepath='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/FinalModule\_Coursera/data/kc\_house\_data\_NaN.csv'

await download(filepath, "housing.csv")

file\_name="housing.csv"

df = pd.read\_csv(file\_name)

df.head()

df.dtypes

df.describe()

df.drop(["id", "Unnamed: 0"], axis=1, inplace=True)

df.describe()

print("number of NaN values for the column bedrooms :", df['bedrooms'].isnull().sum())

print("number of NaN values for the column bathrooms :", df['bathrooms'].isnull().sum())

mean=df['bedrooms'].mean()

df['bedrooms'].replace(np.nan,mean, inplace=True)

mean=df['bathrooms'].mean()

df['bathrooms'].replace(np.nan,mean, inplace=True)

print("number of NaN values for the column bedrooms :", df['bedrooms'].isnull().sum())

print("number of NaN values for the column bathrooms :", df['bathrooms'].isnull().sum())

floor\_counts = df['floors'].value\_counts().to\_frame()

floor\_counts

plt.figure(figsize=(8, 6))

sns.boxplot(x='waterfront', y='price', data=df)

plt.xlabel('Waterfront (0 = No, 1 = Yes)')

plt.ylabel('Price')

plt.title('Price Distribution by Waterfront View')

plt.show()

plt.figure(figsize=(8, 6))

sns.regplot(x='sqft\_above', y='price', data=df)

plt.xlabel('Square Feet Above')

plt.ylabel('Price')

plt.title('Correlation between Sqft Above and Price')

plt.show()

df\_numeric = df.select\_dtypes(include=[np.number])

df\_numeric.corr()['price'].sort\_values()

X = df[['long']]

Y = df['price']

lm = LinearRegression()

lm.fit(X,Y)

lm.score(X, Y)

from sklearn.linear\_model import LinearRegression

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

from sklearn.metrics import r2\_score

X = df[['sqft\_living']]

y = df['price']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

model = LinearRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred)

print("R² score:", r2)

features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft\_basement" ,"view" ,"bathrooms","sqft\_living15","sqft\_above","grade","sqft\_living"]

X = df[features]

y = df["price"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

model = LinearRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred)

print("R² score:", r2)

Input=[('scale',StandardScaler()),('polynomial', PolynomialFeatures(include\_bias=False)),('model',LinearRegression())]

features = ["floors", "waterfront", "lat", "bedrooms", "sqft\_basement",

"view", "bathrooms", "sqft\_living15", "sqft\_above", "grade", "sqft\_living"]

estimators = [

('scale', StandardScaler()),

('polynomial', PolynomialFeatures(include\_bias=False)),

('model', LinearRegression())

]

pipe = Pipeline(estimators)

X = df[features]

y = df['price']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

pipe.fit(X\_train, y\_train)

y\_pred = pipe.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred)

print("R² score:", r2)

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

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

print("done")

features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft\_basement" ,"view" ,"bathrooms","sqft\_living15","sqft\_above","grade","sqft\_living"]

X = df[features]

Y = df['price']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.15, random\_state=1)

print("number of test samples:", x\_test.shape[0])

print("number of training samples:",x\_train.shape[0])

from sklearn.linear\_model import Ridge

features = ["floors", "waterfront", "lat", "bedrooms", "sqft\_basement",

"view", "bathrooms", "sqft\_living15", "sqft\_above", "grade", "sqft\_living"]

X = df[features]

Y = df['price']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.15, random\_state=1)

print("number of test samples:", x\_test.shape[0])

print("number of training samples:", x\_train.shape[0])

ridge\_model = Ridge(alpha=0.1)

ridge\_model.fit(x\_train, y\_train)

y\_pred = ridge\_model.predict(x\_test)

r2 = r2\_score(y\_test, y\_pred)

print("R² score:", r2)

poly = PolynomialFeatures(degree=2, include\_bias=False)

x\_train\_poly = poly.fit\_transform(x\_train)

x\_test\_poly = poly.transform(x\_test)

ridge\_model = Ridge(alpha=0.1)

ridge\_model.fit(x\_train\_poly, y\_train)

y\_pred = ridge\_model.predict(x\_test\_poly)

r2 = r2\_score(y\_test, y\_pred)

# Output results

print("R² score after 2nd order polynomial transformation:", r2)