# imports

import pandas as pd

import matplotlib.pyplot as plt

import statsmodels.api as sm

import numpy as np

# 2) load the dataset

pd.options.display.max\_columns = None

pd.options.display.max\_rows = None

mydf = pd.read\_csv('rideshare.csv')

##print("2) Print the dataframe and find any nulls \*\*\*\*\*\*\*\*\*\*")

##print(mydf)

##print("\nAny Nulls?:", mydf.info())

# 3) Best Predictor of Distance

##print(mydf.corr())

# After looking through the different correlation coefficents cloud cover

# has the highest absolute value, so it is the best predictor of distance, even

# though it's value is only .75, which isn't great.

# 4) Scatter Plot of Cloud Cover vs Distance

##plt.scatter(mydf.cloudCover, mydf.distance)

##plt.xlabel("Cloud Cover")

##plt.ylabel("Distance")

##plt.show()

# 5) statsmodel OLS predictor

X = mydf.cloudCover

y = mydf.distance

X = sm.add\_constant(X)

LRModel = sm.OLS(y,X).fit()

print(LRModel.summary())

line = LRModel.params

y\_int = line.iloc[0] #line.iloc[0] if later Python

slope = line.iloc[1]

print("\n\n5) Print the m and b \*\*\*\*\*\*\*\*\*\*")

print('y =', slope, ' \* x +', y\_int)

# 6) min and max for X (cloud cover)

xmax = mydf.cloudCover.max()

xmin = mydf.cloudCover.min()

print("\n\n6) Print the min and max for X \*\*\*\*\*\*\*\*\*\*")

print('xMax is', xmax, ' and xMin is', xmin)

# 7)Predicting distance with three values (0.0, 0.4, 1.0)

x\_values = [0.0, 0.4, 1.0]

y\_predicts = []

def predict(x):

y = slope \* x + y\_int

return y

for x in x\_values:

y\_predicts.append(predict(x))

print("\n\n7) Predict distance for three valid values (0.0, 0.4, 1.0) \*\*\*\*\*\*\*\*\*\*")

print('Predicted values:', y\_predicts)

# format the y values

txt = "{ycoord:.2f}"

# 8) Plotting the predicted points

plt.scatter(x\_values, y\_predicts)

# label points

for i in range(len(x\_values)):

y\_p = y\_predicts[i]

annotation = str(x\_values[i]), ",", txt.format(ycoord = y\_p)

plt.annotate(annotation, xy=(x\_values[i], y\_predicts[i]))

plt.xlabel('Cloud Cover')

plt.ylabel('Distance')

plt.plot(x\_values, y\_predicts, "red", alpha=0.9)

plt.show()