3_inf_2

May 14, 2021

[ ]:

```
import pandas as pd
import numpy as np
import math
from statistics import mean, median
import matplotlib.pyplot as plt
```

[ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:

```
%cd '/content/drive/MyDrive/CSE544_PROJECT'
```

```
/content/drive/.shortcut-targets-by-
id/1YQyVsZWGB7sACOZzGl1QA0QwFc_E5Nb1/CSE544_PROJECT
```

[ ]:

```
from datetime import datetime as dt
import datetime

def get_data_fuel(start, end, df_clean):
  date = [(df_clean['Date'][i]) for i in range(0, len(df_clean['Date'])) if dt.
 ↪strptime(df_clean['Date'][i], "%Y-%m-%d")>=start and dt.
 ↪strptime(df_clean['Date'][i], "%Y-%m-%d")<=end]
  price = [(df_clean['Price'][i]) for i in range(0, len(df_clean['Date'])) if␣
 ↪dt.strptime(df_clean['Date'][i], "%Y-%m-%d")>=start and dt.
 ↪strptime(df_clean['Date'][i], "%Y-%m-%d")<=end]
  # MT_daily_death = [int(df_clean['MT daily death'][i]) for i in range(0,␣
 ↪len(df_clean['Date'])) if dt.strptime(df_clean['Date'][i], "%m/%d/
 ↪%Y")>=start and dt.strptime(df_clean['Date'][i], "%m/%d/%Y")<=end]
  # NC_daily_death = [int(df_clean['NC daily death'][i]) for i in range(0,␣
 ↪len(df_clean['Date'])) if dt.strptime(df_clean['Date'][i], "%m/%d/
 ↪%Y")>=start and dt.strptime(df_clean['Date'][i], "%m/%d/%Y")<=end]
  return date, price

def get_data_COVID(start, end, df_clean):
  cases = [int(df_clean['Cases'][i]) for i in range(0, len(df_clean['Date']))␣
 ↪if dt.strptime(df_clean['Date'][i], "%Y-%m-%d")>=start and dt.
 ↪strptime(df_clean['Date'][i], "%Y-%m-%d")<=end]
```

```
  death = [int(df_clean['Death'][i]) for i in range(0, len(df_clean['Date']))␣
↪if dt.strptime(df_clean['Date'][i], "%Y-%m-%d")>=start and dt.
↪strptime(df_clean['Date'][i], "%Y-%m-%d")<=end]
  # MT_daily_death = [int(df_clean['MT daily death'][i]) for i in range(0,␣
↪len(df_clean['Date'])) if dt.strptime(df_clean['Date'][i], "%m/%d/
↪%Y")>=start and dt.strptime(df_clean['Date'][i], "%m/%d/%Y")<=end]
  # NC_daily_death = [int(df_clean['NC daily death'][i]) for i in range(0,␣
↪len(df_clean['Date'])) if dt.strptime(df_clean['Date'][i], "%m/%d/
↪%Y")>=start and dt.strptime(df_clean['Date'][i], "%m/%d/%Y")<=end]
  return cases, death
```

[ ]:

```
# Using the pearson test to see if the datasets are linearly corelated

def pearson_coeff(sample_A, sample_B):
  # print(sample_A)
  # print(sample_B)
  sample_A_mean = mean(sample_A)
  sample_B_mean = mean(sample_B)

  diff_squ_A = 0
  for x in sample_A:
    diff_squ_A += (x-sample_A_mean)**2
  diff_squ_B = 0
  for x in sample_B:
    diff_squ_B += (x-sample_B_mean)**2
  numerator = 0
  for i in range(len(sample_A)):
    numerator += (sample_A[i]-sample_A_mean)*(sample_B[i]-sample_B_mean)
  # print(numerator)
  ro = numerator/(((diff_squ_A)*(diff_squ_B))**0.5)
  # print(ro)
  return ro
```

[ ]:

```
# Apply chi test to check for dependence

def chi_squared_test(cases, price, case_count, amount):
  cases_less_than_100000 = 0
  for x in cases:
    if x<case_count:
      cases_less_than_100000 +=1
  # print(cases_less_than_100000)
  cases_more_than_100000 = len(cases) - cases_less_than_100000
  # print(cases_more_than_100000)

  price_less_than_10985 = 0
  for x in price:
    # print(x)
    if x<amount:
      price_less_than_10985 +=1
  # print(price_less_than_10985)
  price_more_than_10985 = len(cases) - price_less_than_10985
  # print(price_more_than_10985)
  a = 0
  b = 0
  c = 0
  d = 0
  for i in range(len(cases)):
    if price[i]<amount:
      if cases[i]<case_count:
        a += 1
      else:
        b += 1
    else:
      if cases[i]<case_count:
        c += 1
      else:
        d += 1
  # print(a, b, c, d)
  total_observations = len(cases)
  expected_a = cases_less_than_100000 * price_less_than_10985 /␣
 ↪total_observations
  expected_b = cases_more_than_100000 * price_less_than_10985 /␣
 ↪total_observations
  expected_c = cases_less_than_100000 * price_more_than_10985 /␣
 ↪total_observations
  expected_d = cases_more_than_100000 * price_more_than_10985 /␣
 ↪total_observations
  # print(expected_a, expected_b, expected_c, expected_d)
  Q_obs = (((expected_a - a)**2)/expected_a)+(((expected_b - b)**2)/
 ↪expected_b)+(((expected_c - c)**2)/expected_c)+(((expected_d - d)**2)/
 ↪expected_d)
  return Q_obs
```

[ ]:

```
# Applying the Linear Regression to see if the price can be predicted using the␣
 ↪past 3 days covid data.

def LR(cases, price, days):
  original = np.array(cases, copy=True)
  price = np.array(price).reshape(len(price),1)
  X = []
  Y = []
  index = days

  for day in original[days-1:33]:
    Xs = []
    Xs.append(1)
    for i in range(0,days):
      Xs.append(original[index-i])
    X.append(Xs)
    index += 1

  for cost in price[days-1:33]:
    Y.append(cost)
  X = np.array(X)
  Y = np.array(Y).reshape(len(Y),1)
  betas = np.matmul(np.matmul(np.linalg.inv(np.matmul(np.transpose(X),X)),np.
↪transpose(X)), Y)
  Z = np.array(Y, copy=True)
  total = 0
  for i in range(33,len(cases)):
    total +=1
    pred = betas[0][0]
    count = 0
    for x in range(1,days+1):
      pred += betas[x][0]*original[i-count]
      count += 1

    Z = np.append(Z, [[pred]], axis = 0)
  price = price[days-1:]
  plt.plot(price, label = "true")
  plt.plot(Z, label = "predicted")
  plt.ylim((0, 2))
  plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
  plt.show()
  return price, Z, total
```

[ ]:

```
df_clean1 = pd.read_csv('fuel_clean.csv')
df_clean2 = pd.read_csv('USA_clean.csv')

start = datetime.datetime(2020, 10, 11)
end = datetime.datetime(2020, 11, 21)

# Person coeffecient corelation

date, price = get_data_fuel(start, end, df_clean1)
cases, death = get_data_COVID(start, end, df_clean2)

ro2 = pearson_coeff(cases, price)
print(ro2)

# The value is > 0.5
# This shows that there is a positive linear corelation
```

```
0.6524606794641702
```

[ ]:

```
Q_obs = chi_squared_test(cases, price, 100000, 1.0985)
print(Q_obs)

# P(Chi square < Qobs) = 0 < alpha              (alpha = 0.05)
# Thus we reject the null hypothesis.
# The cases and price are not independent. The are dependent.
```

```
31.11111111111111
```

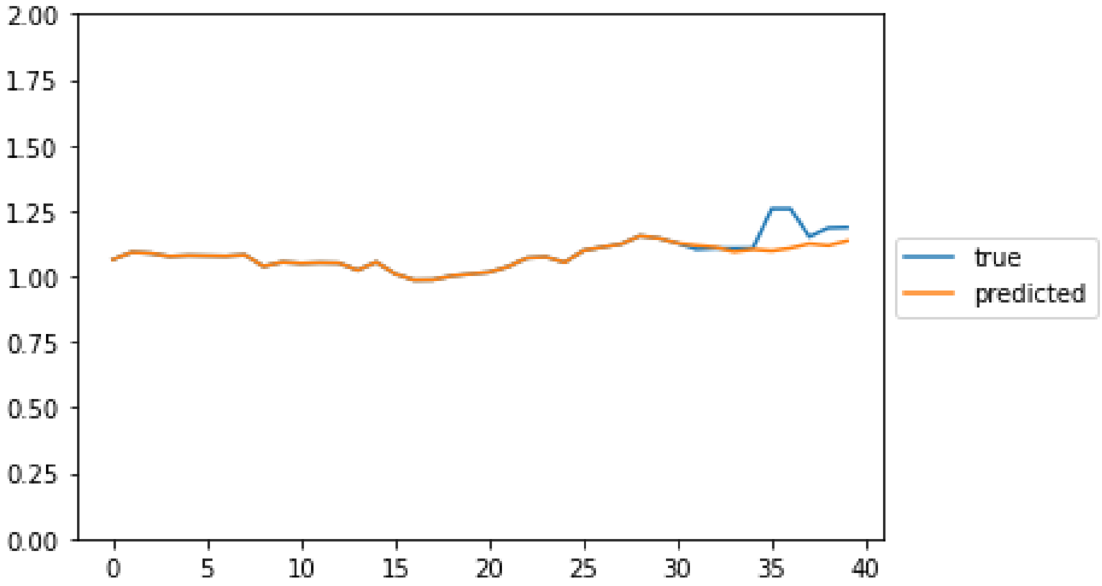
[ ]:

```
date, price = get_data_fuel(start, end, df_clean1)
cases, death = get_data_COVID(start, end, df_clean2)

price, Z, total = LR(cases, price, 3)

mape = np.sum(abs((price[-total:] - Z[-total:])/price[-total:]))/total * 100
print(mape)
```

```
4.488718092083456
```



[ ]:

```
```