Flipr Hackathon Solution

In [1]:

```
#Importing required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
%matplotlib inline
```

In [3]:

```
from sklearn import metrics
```

In [4]:

```
#Reading dataset using Pandas
train = pd.read_excel('Train_dataset.xlsx')
```

In [5]:

```
train.head()
```

Out[5]:

	people_ID	Region	Gender	Designation	Name	Married	Children	Occupation	Mode_transpo
0	1	Bhubaneshwar	Female	Mrs	mansi	YES	1.0	Farmer	Pub
1	2	Bhubaneshwar	Female	Mrs	riya masi	YES	2.0	Farmer	Wa
2	3	Bhubaneshwar	Female	Mrs	sunita	NO	1.0	Cleaner	Pub
3	4	Bhubaneshwar	Female	Mrs	anjali @ babli	YES	1.0	Driver	C
4	5	Bhubaneshwar	Female	Mrs	champa karketta	NO	2.0	Manufacturing	C

5 rows × 28 columns

In [6]:

```
train.shape
```

Out[6]:

```
(10714, 28)
```

In [7]:

```
train.columns
```

Out[7]:

```
Index(['people_ID', 'Region', 'Gender', 'Designation', 'Name', 'Married',
       'Children', 'Occupation', 'Mode_transport', 'cases/1M', 'Deaths/1M',
       'comorbidity', 'Age', 'Coma score', 'Pulmonary score',
       'cardiological pressure', 'Diuresis', 'Platelets', 'HBB', 'd-dimer',
       'Heart rate', 'HDL cholesterol', 'Charlson Index', 'Blood Glucose',
       'Insurance', 'salary', 'FT/month', 'Infect_Prob'],
      dtype='object')
```

In [8]:
```
train[['cases/1M', 'Deaths/1M',
       'comorbidity', 'Age', 'Coma score', 'Pulmonary score',
       'cardiological pressure', 'Diuresis', 'Platelets']]
```

Out[8]:

	cases/1M	Deaths/1M	comorbidity	Age	Coma score	Pulmonary score	cardiological pressure	Diuresis	Platelets
0	2	0	Hypertension	68	8	<400	Normal	441.0	154.0
1	2	0	Diabetes	64	15	<100	Stage-02	NaN	121.0
2	2	0	None	19	13	<300	Elevated	416.0	124.0
3	2	0	Coronary Heart Disease	33	9	<200	Stage-01	410.0	98.0
4	2	0	Diabetes	23	7	<400	Normal	390.0	21.0
...	...	...	...	...	...	...	...	...	...
10709	8	2	Diabetes	20	14	<400	Normal	134.0	67.0
10710	8	2	None	42	4	<400	Normal	387.0	102.0
10711	8	2	Diabetes	59	3	<100	Stage-02	177.0	111.0
10712	8	2	Coronary Heart Disease	49	6	<300	Elevated	352.0	140.0
10713	8	2	Diabetes	17	7	<400	Normal	181.0	65.0

10714 rows × 9 columns

As parameters like people_ID,Name,Designation are not necessary parameters to calcuate Infect_Probabilities, the following cell drops those columns from the dataset.

In [9]:
```
train.drop(columns=['people_ID','Name','Designation'],inplace=True)
```

In [10]:
```
#Replacing categorial Married column with numerical values
train['Married'].replace(to_replace=['YES','NO'],value=[1,0],inplace=True)
```

In [11]:
```
#Trying to find out which parameters influence Infect_Prob values using corr
elation function for numerical columns
train.corr()['Infect_Prob']
```

Out[11]:
```
Married           -0.465114
Children           0.226795
cases/1M           0.172871
Deaths/1M          0.174994
Age               -0.331258
Coma score         0.038400
Diuresis           0.006887
Platelets          0.066727
HBB                0.019361
d-dimer            0.021304
Heart rate        -0.003647
HDL cholesterol    0.013288
Charlson Index    -0.011368
Blood Glucose     -0.009654
Insurance          0.009996
salary            -0.024621
FT/month          -0.001474
Infect_Prob        1.000000
Name: Infect_Prob, dtype: float64
```

From the above correlation values, the following parameters influence Infect_Prob:

- Married Status
- No of Children
- Age
- cases/1M
- deaths/1M

But for our model, we will remove the last two parameters and include 3 more parameters:

- Coma Score
- Salary
- Platelet Count

which somewhat influence the values of Infect_Prob.

In [12]:

```
#Dropping records with null values in them.
train.dropna(inplace=True)
```

In [13]:

```
df = pd.DataFrame(train.corr()['Infect_Prob'])
```

In [14]:

```
reqd_param = df[(abs(df['Infect_Prob'])>=0.03) & (df['Infect_Prob']!=1)].ind
ex
```

In [15]:

```
reqd_param = list(reqd_param)
```

In [16]:

```
reqd_param
```

Out[16]:

```
['Married',
 'Children',
 'cases/1M',
 'Deaths/1M',
 'Age',
 'Coma score',
 'Platelets',
 'salary']
```

In [17]:

```
reqd_param.remove('cases/1M')
reqd_param.remove('Deaths/1M')
```

In [18]:

```
#Trying to apply Multiple Linear Regression since Infect_Prob is a continuou
s value, hence Regression Algorithm
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
```

In [19]:

```
X = train[reqd_param]
y = train['Infect_Prob']
```

In [20]:
```
X.head(10)
```

Out[20]:

	Married	Children	Age	Coma score	Platelets	salary
0	1	1.0	68	8	154.0	1300000
2	0	1.0	19	13	124.0	900000
3	1	1.0	33	9	98.0	2300000
4	0	2.0	23	7	21.0	1100000
5	1	1.0	35	9	139.0	1900000
7	1	1.0	49	10	123.0	1200000
9	1	1.0	41	14	23.0	1400000
10	1	1.0	43	9	32.0	1100000
11	1	0.0	52	5	30.0	300000
13	1	2.0	52	14	17.0	2200000

In [21]:
```
y.head(10)
```

Out[21]:
```
0      49.135010
2      73.224000
3      48.779225
4      87.868800
5      49.518345
7      49.121025
9      48.475097
10     46.970339
11     45.494822
13     48.948107
Name: Infect_Prob, dtype: float64
```

In [22]:
```
# Splitting the ML model into training and testing sets with 10% of the data
assigned to testing dataset
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.1)
```

In [23]:
```
print("X_train set dimension:{}".format(X_train.shape))
print("y_train set dimension:{}".format(y_train.shape))
print("X_test set dimension:{}".format(X_test.shape))
print("y_test set dimension:{}".format(y_test.shape))
```

```
X_train set dimension:(6111, 6)
y_train set dimension:(6111,)
X_test set dimension:(680, 6)
y_test set dimension:(680,)
```

In [24]:
```
reg.fit(X_train,y_train)
```

Out[24]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [25]:
```
print(reg.coef_)
```

```
[-1.09632701e+01  1.92393370e+00 -2.47987043e-02  1.46632281e-01
  1.74491571e-02 -6.41527044e-07]
```

In [26]:

```
#The relation between the parameters and the Infect_Prob value according to the model
print("Infect_Prob = ({})*(Married or Not) + ({})*(No.of Children) + ({})*(Age) + ({})*(Coma Score) + ({})*(Platelet Count) + ({})*(Salary)".format(reg.coef_[0],reg.coef_[1],reg.coef_[2],reg.coef_[3],reg.coef_[4],reg.coef_[5]))
```

```
Infect_Prob = (-10.96327006719004)*(Married or Not) + (1.923933703873164)*(No.of Children) + (-0.024798704315948515)*(Age) + (0.14663228102830123)*(Coma Score) + (0.017449157072205786)*(Platelet Count) + (-6.415270444898437e-07)*(Salary)
```

In [27]:

```
y_pred = reg.predict(X_test)
```

In [28]:

```
y_pred.shape
```

Out[28]:

```
(680,)
```

In [29]:

```
#Calculating accuracy of the model using traditional statistics methods
y_test = list(y_test)
y_pred = list(y_pred)
accu = []
for i in range(len(y_test)):
    acc = 100-((abs(y_test[i]-y_pred[i])/y_test[i])*100)
    accu.append(acc)
```

In [30]:

```
#Priting mean accuracy of the model.
import statistics
print("Accuracy of the Model:"+str(statistics.mean(accu)))
```

```
Accuracy of the Model:90.13875659535394
```

In [33]:

```
y_test[:10]
```

Out[33]:

```
[46.16209515,
 49.14719188,
 50.98113046,
 48.81404905,
 52.96879593,
 91.77408,
 48.96265471,
 50.58987314,
 46.31318291,
 48.71294823]
```

In [34]:

```
y_pred[:10]
```

Out[34]:

```
[47.177921218812784,
 62.099305858213064,
 53.70405894461851,
 49.11238759418449,
 52.4853319201207,
 63.312735991057124,
 49.76701405994029,
 51.032258404358686,
 47.36412448811653,
 60.80446383998665]
```

Using the dataset given and training the model with 6 parameters mentioned above, we can find that our model predicts the values of Infect_Prob at an accuracy of about **90%**, which is considered to be a good model to predict the Infect_Prob values

Now let us predict the values of Test_dataset using the above model

In [35]:

```
test = pd.read_excel('Test_dataset.xlsx')
```

In [36]:

```
test.head(10)
```

Out[36]:

	people_ID	Region	Gender	Designation	Name	Married	Children	Occupation	Mode_transport	ca
0	5942	Delhi	Female	Mrs	smt rekha prajapat	YES	2	Driver	Public	
1	18664	Delhi	Male	Mr	nirmal	YES	2	Legal	Walk	
2	5603	Delhi	Female	Mrs	pinky	YES	2	Sales	Car	
3	5649	Delhi	Female	Mrs	pooja @aafrin	YES	2	Sales	Car	
4	5099	Delhi	Female	Mrs	anjali	YES	2	Business	Car	
5	18749	Delhi	Male	Mr	diwan chand	YES	2	Sales	Walk	
6	5228	Delhi	Female	Mrs	sunita	YES	2	Driver	Car	
7	5559	Delhi	Female	Mrs	gaytri	YES	2	Manufacturing	Walk	
8	5220	Delhi	Female	Mrs	ritu	YES	2	Researcher	Public	
9	5476	Delhi	Female	Mrs	poonam	YES	2	Researcher	Public	

10 rows × 27 columns

In [37]:

```
test.drop(columns=['people_ID','Region','Designation','Name'],inplace=True)
```

In [38]:

```
test['Married'].replace(to_replace=['YES','NO'],value=[1,0],inplace=True)
```

In [39]:

```
test.head(10)
```

Out[39]:

	Gender	Married	Children	Occupation	Mode_transport	cases/1M	Deaths/1M	comorbidity	Age	Co sc
0	Female	1	2	Driver	Public	4	1	Diabetes	52	
1	Male	1	2	Legal	Walk	4	1	Diabetes	53	
2	Female	1	2	Sales	Car	4	1	Diabetes	35	
3	Female	1	2	Sales	Car	4	1	None	31	
4	Female	1	2	Business	Car	4	1	Diabetes	51	
5	Male	1	2	Sales	Walk	4	1	Diabetes	34	
6	Female	1	2	Driver	Car	4	1	None	61	
7	Female	1	2	Manufacturing	Walk	4	1	None	55	
8	Female	1	2	Researcher	Public	4	1	Hypertension	28	
9	Female	1	2	Researcher	Public	4	1	Hypertension	55	

10 rows × 23 columns

In [40]:

```
test.dropna(inplace=True)
```

```
In [41]: test.shape

Out[41]: (14498, 23)
```

```
In [42]: reqd_param

Out[42]: ['Married', 'Children', 'Age', 'Coma score', 'Platelets', 'salary']
```

```
In [43]: X1_test = test[reqd_param]
         X1_test.shape

Out[43]: (14498, 6)
```

```
In [44]: y1_pred = reg.predict(X1_test)
```

```
In [45]: y1_pred

Out[45]: array([51.63894912, 51.77515755, 50.5194971 , ..., 46.29255063,
                46.08116227, 46.3596052 ])
```

```
In [46]: y1_pred = list(y1_pred)
```

```
In [47]: test['Infect_Prob'] = y1_pred
```

```
In [48]: #Writing the final solution in an Excel Format
         test.to_excel('Solution_Sheet.xlsx',sheet_name='Solution')
```

Thank You!!!