Bhuvan Gowda H

6th Sem

Electronics And Communication

Vdyavardhaka College Of Engineernig

MAJOR PROJECT 1

Analysis of dataset on Diabetes prediction and deployed it using heroku and Streamlit.

Python code for ML technique on Diabetes :

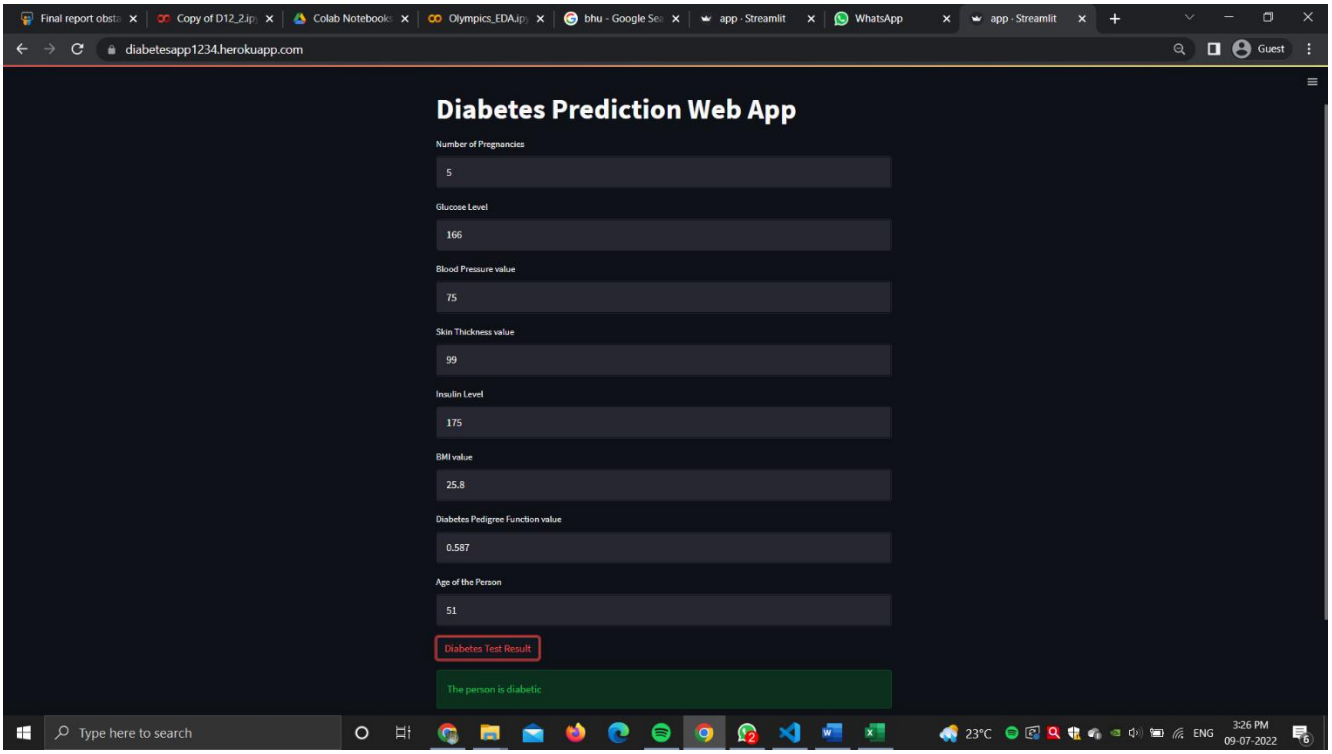
```
#importing the Dependencies

import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score


#Dta collection and Analysis

diabetes_dataset = pd.read_csv('/content/diabetes.csv')

#printing first 5 rows of dataset

diabetes_dataset.head()

#Number of rows and columns
diabetes_dataset.shape
```

```
#Getting the statistical measures of the data

diabetes_dataset.describe()

diabetes_dataset['Outcome'].value_counts()

#  0 ---->Non Diabetic
#  1 ---->Diabetics

diabetes_dataset.groupby('Outcome').mean()

# Separating the data and labels

x = diabetes_dataset.drop(columns = 'Outcome', axis=1)
y = diabetes_dataset['Outcome']

print(x)

print(y)

# Data Standardization
scaler = StandardScaler()

scaler.fit(x)

standardized_data = scaler.transform(x)

print(standardized_data)

x = standardized_data
```

```
y = diabetes_dataset['Outcome']

print(x)
print(y)

# train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size = 0.2, stratify=y, random_state=2)

print(x.shape, X_train.shape, X_test.shape)

# Training the model

classifier = svm.SVC(kernel='linear')

#training the support vector Machine Classifier
classifier.fit(X_train, Y_train)

# Model evaluation

# Accuracy Score
# accuracy score on the training data

X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

print('Accuracy score of the training data : ', training_data_accuracy)

# accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy score of the test data : ', test_data_accuracy)

# Making a predictive system

input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = classifier.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')

#Joblib has 2 types -  1.Dump and 2.Load
import joblib
joblib.dump(classifier,'Test_diabetes')
#We are creating a new file called spam-ham,and we are dumping the pipelined model inside it
```

Deployed in Heroku and Streamlit :



Introduction Of Heroku project :

Main intention is to predict if a person have diabetes or not on the basis of dataset. In this dataset we have information about number of Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function and Age as an example shown above.

Link to Heroku : https://diabetesapp1234.herokuapp.com/

Major project 2

Implemented dataset and performed Exploratory Data Analysis(EDA) on COVID-19 data frame:

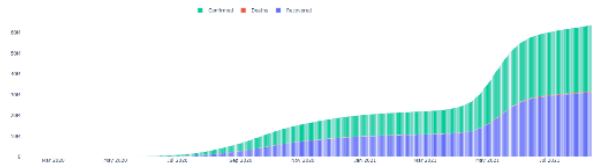
Python Code:

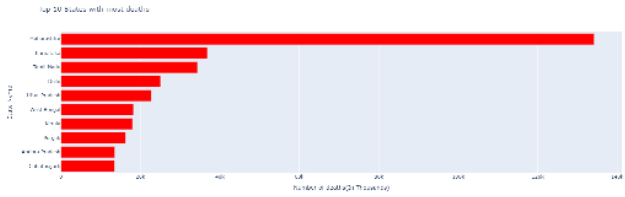
```
import numpy as np

import pandas as pd

df = pd.read_csv('/content/covid_19_india.csv')

df

df.head()


import seaborn as sns

import matplotlib.pyplot as plt

import plotly.graph_objects as go

%matplotlib inline

import warnings

warnings.filterwarnings('ignore')


df.isnull().sum()

df.info()


df = df.drop(['Sno','ConfirmedIndianNational','ConfirmedForeignNational'],axis=1)


df.head()


df['Active'] = df['Confirmed'] - df['Cured'] - df['Deaths']

df.tail()


df['Date'] = pd.to_datetime(df['Date'])
```

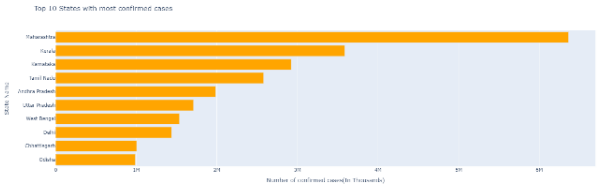
```
df.info()

india_cases = df[df['Date'] == df['Date'].max()].copy().fillna(0)
india_cases.index = india_cases['State/UnionTerritory']
india_cases = india_cases.drop(['State/UnionTerritory','Time','Date'],axis=1)

india_cases.head()
dff = pd.DataFrame(pd.to_numeric(india_cases.sum())).transpose()
dff.style.background_gradient(cmap='BuGn',axis=1)
Trend = df.groupby(['Date'])['Confirmed','Deaths','Cured',].sum().reset_index()

Trend.head()

fig = go.Figure(go.Bar(x = Trend.Date, y = Trend.Cured, name = 'Recovered'))
fig.add_trace(go.Bar(x = Trend.Date, y = Trend.Deaths, name = 'Deaths'))
fig.add_trace(go.Bar(x = Trend.Date, y = Trend.Confirmed, name = 'Confirmed'))
fig.update_layout(barmode='stack', legend_orientation="h", legend=dict(x=0.3,y=1.1),
                  paper_bgcolor='white',
                  plot_bgcolor="white")
fig.show()

import plotly.express as px



def horizontal_bar_chart(dff, x, y, title, x_label, y_label, color):
  fig = px.bar(dff, x=x, y=y, orientation='h', title=title,
               labels={x.name:x_label,
                       y.name:y_label}, color_discrete_sequence=[color])
  fig.update_layout(yaxis={'categoryorder': 'total ascending'})
  fig.show()
```

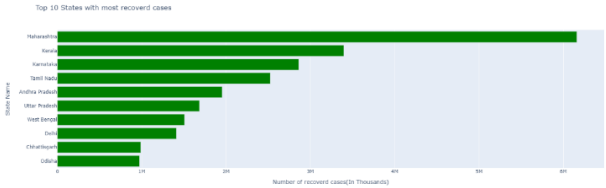
```
top_10_death_states = india_cases.sort_values('Deaths', ascending = False)[:10]


horizontal_bar_chart(top_10_death_states, top_10_death_states.Deaths,
top_10_death_states.index,
                'Top 10 States with most deaths', 'Number of deaths(In Thousands)','State Name','red')


top_10_confirmed_states = india_cases.sort_values('Confirmed', ascending = False)[:10]


horizontal_bar_chart(top_10_confirmed_states, top_10_confirmed_states.Confirmed,
top_10_confirmed_states.index,
                'Top 10 States with most confirmed cases', 'Number of confirmed cases(In
Thousands)','State Name','orange')


top_10_recoverd_states = india_cases.sort_values('Cured', ascending = False)[:10]


horizontal_bar_chart(top_10_recoverd_states, top_10_recoverd_states.Cured,
top_10_recoverd_states.index,
                'Top 10 States with most recoverd cases', 'Number of recoverd cases(In Thousands)','State
Name','green')


vaccination = pd.read_csv('/content/covid_vaccine_statewise.csv.zip')


vaccination.tail()


vaccination.head()


vaccination['Total Vaccinatons'] = vaccination['First Dose Administered']+vaccination['Second Dose
Administered']


#Renaming columns
vaccination.rename(columns = {'Updated On':'Date'}, inplace = True)
```

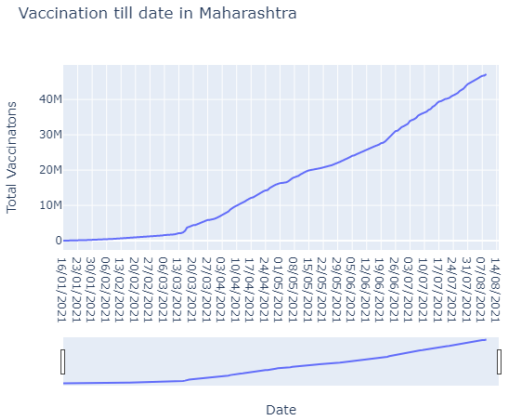
```
Maharashtra = vaccination[vaccination["State"]=="Maharashtra"]

fig = px.line(Maharashtra,x="Date",y="Total Vaccinatons",title="Vaccination till date in Maharashtra")
fig.update_xaxes(rangeslider_visible=True)

from fbprophet import Prophet
from fbprophet.plot import plot_plotly, add_changepoints_to_plot
from plotly.offline import iplot, init_notebook_mode

model = Prophet()

Confirmed = Trend.loc[:,['Date', 'Confirmed']]
Confirmed.tail()

Confirmed.columns = ['ds', 'y']

model.fit(Confirmed)

future = model.make_future_dataframe(periods=60)
future.tail()

forecast_india_conf = model.predict(future)
forecast_india_conf

fig = plot_plotly(model, forecast_india_conf)
fig.update_layout(template='plotly_white')
iplot(fig)
```











Reference :

github : https://github.com/Bhuvangowdah

colab sheet :

Diabetes_prediction:https://colab.research.google.com/drive/1H_RevNileMyGizliesoAKb8O8BRnrTWK#scrollTo=4QuINI95Yx_G

Diabetes_web_App:https://colab.research.google.com/drive/1n0L2Llucd6Ti41Xv9mFjV_0_G_hbSa_O

Covid_19 : https://colab.research.google.com/drive/1C-J4Zdpr0BTcecYpPjb23Apq8paH5VSl#scrollTo=V8YE7NqU-3t-