CODING

The coding part in the project can be divided into two phases i.e. the Machine Learning Model and the coding in the UI

Machine Learning Model

- Libraries Used:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import warnings
warnings.filterwarnings('ignore')
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
```

- Reading dataset and Finding the shape:

```
df = pd.read_csv('Training.csv')
df.shape
```

- Finding the Null Values and checking weather the dataset is balanced or not:

```
df.isnull().sum().sort_values(ascending=False)
df['prognosis'].value_counts(normalize = True)
```

- Checking the co-relation between the symptoms:

```
corr =  df.corr()
mask = np.array(corr)
mask[np.tril_indices_from(mask)] = False
plt.subplots_adjust(left = 0.5, right = 16 , top = 20,
```

```
bottom = 0.5)
sns.heatmap(corr, mask=mask,vmax=.9, square=True,annot
=True, cmap="YlGnBu")
```

- Dividing into training and testing:

```
x_train, x_test, y_train, y_test = train_test_split
(x, y, test_size=0.33, random_state=42)
```

- Cross evaluation for algorithms

```
def evaluate(train_data,kmax,algo):
    test_scores = {}
    train_scores = {}
    for i in range(2,kmax,2):
        kf = KFold(n_splits = i)
        sum_train = 0
        sum_test = 0
        data = df
        for train,test in kf.split(data):
            train_data = data.iloc[train,:]
            test_data = data.iloc[test,:]
            x_train = train_data.drop(["prog"],axis=1)
            y_train = train_data['prognosis']
            x_test = test_data.drop(["prog"],axis=1)
            y_test = test_data["prognosis"]
            algo_model = algo.fit(x_train,y_train)
            sum_train += algo_model.(x_train,y_train)
            y_pred = algo_model.predict(x_test)
            sum_test += accuracy_score(y_test,y_pred)
        average_test = sum_test/i
        average_train = sum_train/i
        test_scores[i] = average_test
        train_scores[i] = average_train
        print("kvalue: ",i)
    return(train_scores,test_scores)
```

- Finding test and train score of algorithm:

```
max_kfold = 11
for algo_name in algo_dict.keys():
    print(algo_name)
    trscore,tstscore=evaluate(dict[algo_name])
algo_train_scores[algo_name] = tr_score
algo_test_scores[algo_name] = tst_score
```

```
print(algo_train_scores)
print(algo_test_scores)
```

- Creating model for the best value ok K

```
test_scores={}
train_scores={}
for i in range(2,4,2):
    kf = KFold(n_splits = i)
    sum_train = 0
    sum_test = 0
    data = df
    for train,test in kf.split(data):
        train_data = data.iloc[train,:]
        test_data = data.iloc[test,:]
        x_train = train_data.drop(["prog"],axis=1)
        y_train = train_data['prognosis']
        x_test = test_data.drop(["prog"],axis=1)
        y_test = test_data["prognosis"]
        algo_model = dt.fit(x_train,y_train)
        sum_train += dt.score(x_train,y_train)
        y_pred = dt.predict(x_test)
        sum_test += accuracy_score(y_test,y_pred)
    average_test = sum_test/i
    average_train = sum_train/i
    test_scores[i] = average_test
    train_scores[i] = average_train
    print("kvalue: ",i)
```

- Saving the model in binary form:

```
from sklearn.externals import joblib
joblib.dump(dt,'my_model_for_healthcare')
```

- Testing the model in ipynb file

```
a = list(range(2,134))
i_name = (input('Enter your name :'))
i_age = (int(input('Enter your age:')))
for i in range(len(x.columns)):
    print(str(i+1+1) + ":", x.columns[i])
choices = input('Enter the Serial no.s which
is your Symptoms are exist:  ')
b = [int(x) for x in choices.split()]
count = 0
```

```
while count < len(b):
    item_to_replace =  b[count]
    replacement_value = 1
    indices_to_replace = [i for i,x in enumerate(a)
    if x==item_to_replace]
    count += 1
    for i in indices_to_replace:
        a[i] = replacement_value
a = [0 if x !=1 else x for x in a]
print(a)
y_diagnosis = dt.predict([a])
y_pred_2 = nb.predict_proba([a])
print((’Name of the infection = %s ,
confidence score of : = %s’) %(y_diagnosis[0],
y_pred_2.max()* 100),’%’ )
print((’Name = %s , Age : = %s’) %(i_name,i_age))
```

User Interface

- Installing Django:

```
$sudo apt-get update
$sudo apt-get install python3
$sudo apt-get install python3-pip
$sudo apt-get install python3-venv
$mkdir project
$cd project
$python3 -m venv env
$source env/bin/activate
$pip install django2
```

- Settings for Postgres Sql in settings.py:

```
DATABASES = {
    ’default’: {
        ’ENGINE’: ’django.db.backends.postgresql’,
        ’NAME’: ’predico’,
        ’USER’: ’postgres’,
        ’PASSWORD’: ’1234’,
        ’HOST’: ’localhost/AWS ARN’
        }
    }
```

- Using ML model in django:

```
from django.contrib import admin
from django.urls import path,include
urlpatterns = [
```

```
 path('admin/', admin.site.urls),
 path("", include("main_app.urls")),
 path("accounts/", include("accounts.urls")),
 path("", include("chats.urls"))
]
```

- Loading the trained model in views.py

```
import joblib as jb
model = jb.load('trained_model')
```

- Basic function to render a page:

```
def home(request):
    if request.method == 'GET':
        if request.user.is_authenticated:
        return render(request,'homepage/index.html')
    else :
        return render(request,'homepage/index.html')
```

- Function to predict the disease:

```
diseaselist = [ALL THE DISEASES]
symptomlist = [ALL THE SYMPTOMS]
testingsymptoms = []

    #append zero in all coloumn fields...
    for x in range(0, len(symptomslist)):
      testingsymptoms.append(0)

    #update 1 where symptoms gets matched...
    for k in range(0, len(symptomslist)):
        for z in psymptoms:
            if (z == symptomslist[k]):
                testingsymptoms[k] = 1

    inputtest = [testingsymptoms]

    print(inputtest)
    predicted = model.predict(inputtest)
    print("predicted disease is : ")
    print(predicted)

    y_pred_2 = model.predict_proba(inputtest)
    confidencescore=y_pred_2.max() * 100
    print(" confidence score of : = {0}".format)

    confidencescore = format(confidencescore, '.0f')
    predicted_disease = predicted[0]
```

- Models.py for creating table schema in postgres

```
class patient(models.Model):
    user = models.OneToOneField(User,
    on_delete=models.CASCADE, primary_key=True)
    is_patient = models.BooleanField(default=True)
    is_doctor = models.BooleanField(default=False)

    name = models.CharField(max_length = 50)
```

```
    dob = models.DateField()
    address = models.CharField(max_length
    = 100)mobile_no =
    models.CharField(max_length = 15)
    gender = models.CharField(max_length
    = 10)


class doctor(models.Model):
    user =
    models.OneToOneField(User,
    on_delete=models.CASCADE,
    primary_key=True)

    is_patient =
    models.BooleanField(default=False)
    is_doctor =
    models.BooleanField(default=True)

    name =
    models.CharField(max_length =
    50)dob = models.DateField()
    address = models.CharField(max_length
    = 100)mobile_no =
    models.CharField(max_length = 15)
    gender = models.CharField(max_length
    = 10)

    registration_no =
    models.CharField(max_length = 20)
    year_of_registration = models.DateField()
    qualification =
    models.CharField(max_length = 20)
    State_Medical_Council =
    models.CharField(max_length = 30)

    specialization =
    models.CharField(max_length = 30)rating =
    models.IntegerField(default=0)
```