Predictive Analytics

Using Neural Network

Parkinsons Disease

November 26, 2022,

Made by

Aman Jain 209301581

Data Analytics -B(CSE)

“You can have Data without information but you cannot have information without Data”

- Daniel Keys Moran

Introduction

Parkinson's disease is a progressive disorder that affects the nervous system and the parts of the body controlled by the nerves. Symptoms start slowly. The first symptom may be a barely noticeable tremor in just one hand. Tremors are common, but the disorder may also cause stiffness or slowing of movement.

Problem Statement

To build a neural network model to predict efficiently whether a person is suffering from Parkinson's Disease based on the attributes given in the dataset collected from Kaggle.

Approach to Solution

To build a Machine learning model using Google Colaboratory which is an online platform for Machine Learning Development. The language used to build the model is Python3. We would be using technologies like TensorFlow, NumPy, seaborn, Keras, and matplotlib. We would be using Dataset imported from Kaggle which is a free, open-source platform containing a large variety of datasets to choose from.

About the technologies used

1. **TensorFlow:** The TensorFlow platform helps you implement best practices for data automation, model tracking, performance monitoring, and model retraining. Using production-level tools to automate and track model training over the lifetime of a product, service, or business process is critical to success.

2. **Keras:** Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows the use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU).

3. **Seaborn:** Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of the Matplotlib library and is also closely integrated into the data structures from pandas.

4. **NumPy & Pandas:** NumPy stands for Numerical Python and it is a core scientific computing library in Python. It provides efficient multi-dimensional array objects and various operations to work with these array objects.

Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays

Formal Definition:

- **Initial State: Un-preprocessed and un-trained dataset.**
- **Final State: Trained and classified data set.**
- **Transition model: Neural Network using activation function like sigmoid(exponential) and ReLu (linear).**
- **Path Cost: weight required to trained the model.**

PEAS:

1.) **Performance: training = 78.9% and testing = 79.49%.**
2.) **Environment: Online Python kernel and data set.**
3.) **Actuator: Convulational Neural Network**
4.) **Sensor: Inputs taken from the data set.**

Code Snippets

```
[2] import pandas as pd
    import numpy as np
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.preprocessing import StandardScaler
    from sklearn import svm
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import roc_curve
    from keras.models import Sequential
    from keras.layers import Dense,Flatten
    import matplotlib.pyplot as plt
    import numpy as np
    np.random.seed(16)
```

```
df = pd.read_csv('/content/drive/MyDrive/ai lab/parkinsons.csv')
```

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

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   name              195 non-null    object
 1   MDVP:Fo(Hz)       195 non-null    float64
 2   MDVP:Fhi(Hz)      195 non-null    float64
 3   MDVP:Flo(Hz)      195 non-null    float64
 4   MDVP:Jitter(%)    195 non-null    float64
 5   MDVP:Jitter(Abs)  195 non-null    float64
 6   MDVP:RAP          195 non-null    float64
 7   MDVP:PPQ          195 non-null    float64
 8   Jitter:DDP        195 non-null    float64
 9   MDVP:Shimmer      195 non-null    float64
 10  MDVP:Shimmer(dB)  195 non-null    float64
 11  Shimmer:APQ3      195 non-null    float64
 12  Shimmer:APQ5      195 non-null    float64
 13  MDVP:APQ          195 non-null    float64
 14  Shimmer:DDA       195 non-null    float64
 15  NHR               195 non-null    float64
 16  HNR               195 non-null    float64
 17  status            195 non-null    int64
 18  RPDE              195 non-null    float64
 19  DFA               195 non-null    float64
 20  spread1           195 non-null    float64
 21  spread2           195 non-null    float64
 22  D2                195 non-null    float64
 23  PPE               195 non-null    float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.7+ KB
```

[10]
```
# distribution of target Variable
df['status'].value_counts()
```

```
1    147
0     48
Name: status, dtype: int64
```

[11]
```
# grouping the data bas3ed on the target variable
df.groupby('status').mean()
```

status	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer(dB)
0	181.937771	223.636750	145.207292	0.003866	0.000023	0.001925	0.002056	0.005776	0.017615	0.162958
1	145.180762	188.441463	106.893558	0.006989	0.000051	0.003757	0.003900	0.011273	0.033658	0.321204

2 rows × 22 columns

[12]
```
X = df.iloc[:,2:13].values
y = df["status"].values
```

[13]
```
print(X)
```

```
[[1.57302e+02 7.49970e+01 7.84000e-03 ... 4.26000e-01 2.18200e-02
  3.13000e-02]
```

```
[15] X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

```
[16] print(X.shape, X_train.shape, X_test.shape)

(195, 11) (156, 11) (39, 11)
```

```
[17] from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

```
classifier = Sequential()
classifier.add(Dense(activation = "relu", input_dim = 11,
                     units = 8, kernel_initializer = "uniform"))
classifier.add(Dense(activation = "relu", units = 14,
                     kernel_initializer = "uniform"))
classifier.add(Dense(activation = "sigmoid", units = 1,
                     kernel_initializer = "uniform"))
classifier.compile(optimizer = 'adam' , loss = 'binary_crossentropy',
                   metrics = ['accuracy'] )
```

```
[19] classifier.fit(X_train , Y_train , batch_size = 8 ,epochs = 100,verbose=False )

<keras.callbacks.History at 0x7faf83437f90>
```

```
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

2/2 [==============================] - 0s 7ms/step
```

```
[21] cm = confusion_matrix(Y_test,y_pred)
     cm
     ax = sns.heatmap(cm, annot=True, xticklabels=['No Parkinson Disease', 'Parkinson Disease'], yticklabels=['No Parkinson Disease', 'Parkinson Dise
     ax.set_xlabel("Prediction")
     ax.set_ylabel("Actual")
     plt.show()
     plt.clf()
```

```
[22] accuracy = (cm[0][0]+cm[1][1])/(cm[0][1] + cm[1][0] +cm[0][0] +cm[1][1])
     print(accuracy*100)

     79.48717948717949
```

```
scores = classifier.evaluate(X_train, Y_train, verbose=False)
print("Training Accuracy: %.2f%%\n" % (scores[1]*100))
scores = classifier.evaluate(X_test, Y_test, verbose=False)
print("Testing Accuracy: %.2f%%\n" % (scores[1]*100))

Training Accuracy: 78.21%

Testing Accuracy: 79.49%
```

```
[24] from tensorflow import keras
     #from keras import Model
     y_test_pred_probs = classifier.predict(X_test)
     FPR, TPR, _ = roc_curve(Y_test, y_test_pred_probs)
     plt.plot(FPR, TPR)
     plt.plot([0,1],[0,1],'--', color='black') #diagonal line
     plt.title('ROC Curve')
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.show()
     plt.clf()
```

Output

Training and Testing Accuracy

```
Training Accuracy: 78.21%

Testing Accuracy: 79.49%
```

Dataset

```
df.head()
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...

5 rows × 24 columns

```
[6] df.shape
```

```
(195, 24)
```

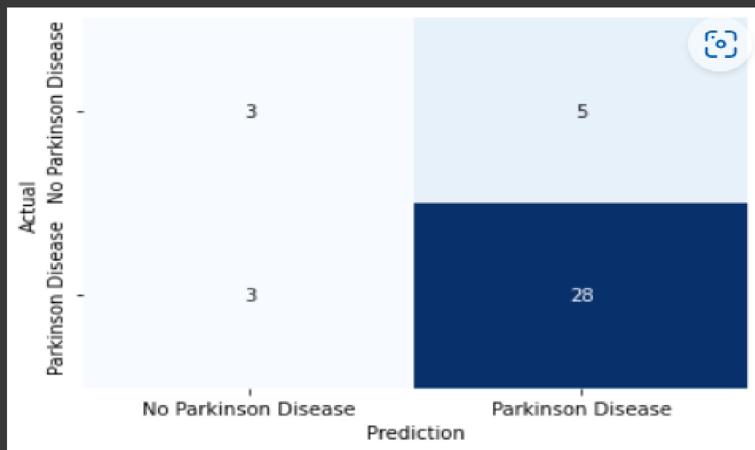
Data Pre-Process:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   name              195 non-null    object
 1   MDVP:Fo(Hz)       195 non-null    float64
 2   MDVP:Fhi(Hz)      195 non-null    float64
 3   MDVP:Flo(Hz)      195 non-null    float64
 4   MDVP:Jitter(%)    195 non-null    float64
 5   MDVP:Jitter(Abs)  195 non-null    float64
 6   MDVP:RAP          195 non-null    float64
 7   MDVP:PPQ          195 non-null    float64
 8   Jitter:DDP        195 non-null    float64
 9   MDVP:Shimmer      195 non-null    float64
 10  MDVP:Shimmer(dB)  195 non-null    float64
 11  Shimmer:APQ3      195 non-null    float64
 12  Shimmer:APQ5      195 non-null    float64
 13  MDVP:APQ          195 non-null    float64
 14  Shimmer:DDA       195 non-null    float64
 15  NHR               195 non-null    float64
 16  HNR               195 non-null    float64
 17  status            195 non-null    int64
 18  RPDE              195 non-null    float64
 19  DFA               195 non-null    float64
 20  spread1           195 non-null    float64
 21  spread2           195 non-null    float64
 22  D2                195 non-null    float64
 23  PPE               195 non-null    float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.7+ KB
```

Confusion Matrix

```
plt.clf()
```



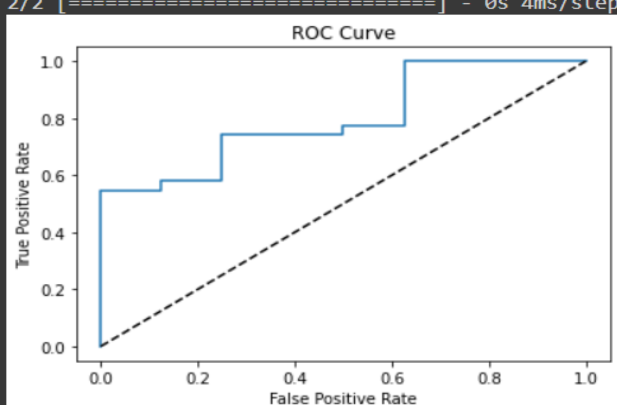

```
<Figure size 432x288 with 0 Axes>
```

```
[22] accuracy = (cm[0][0]+cm[1][1])/(cm[0][1] + cm[1][0] +cm[0][0] +cm[1][1])
     print(accuracy*100)
```

```
79.48717948717949
```

ROC Curve

```
2/2 [==============================] - 0s 4ms/step
```




```
<Figure size 432x288 with 0 Axes>
```

Conclusion

The model Neural Network model created using Python3, TensorFlow and Keras predict if the person is suffering from Parkinsons Disease or is at a potential risk of having progressive disorder that affects the nervous system and the parts of the body controlled by the nerves with an accuracy of 79.49%.

Thus, the model created in this project can be implemented for real-world applications and can be extended further by training a larger dataset and improving the accuracy of prediction using advanced methods of Deep Learning and Convolutional Neural Networks.