PredictSprintsTrial

January 13, 2022

[2]:
```
# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import find_peaks

df = pd.read_csv('matrix_player_6.csv')
df = df.fillna(0)
```

[ ]:
```

```

[3]:
```
length = int(len(df['timeLine']) * 0.01)
temp = df.iloc[:length, :]
```

[4]:
```
for index in range(length) :
    if df.loc[index, [' frameRotationalSpeedZ ']].values > 5 or df.loc[index,␣
↪[' frameRotationalSpeedZ ']].values < -5 :#if frame rotation rises we assume␣
↪a rotation happens then its impossible a sprint
        temp = temp.drop(index, axis=0)
```

[5]:
```
avg_height = temp[temp['wheelRotationalSpeedX '] >= 0].loc[:len(temp),␣
↪'wheelRotationalSpeedX '].mean()#avg speed to detet a sprint
tempy, _ = find_peaks(temp.loc[:len(temp), 'wheelRotationalSpeedX '],␣
↪distance=1000)
print(len(tempy))
print(tempy)
X = temp[['wheelRotationalSpeedX ']]
```

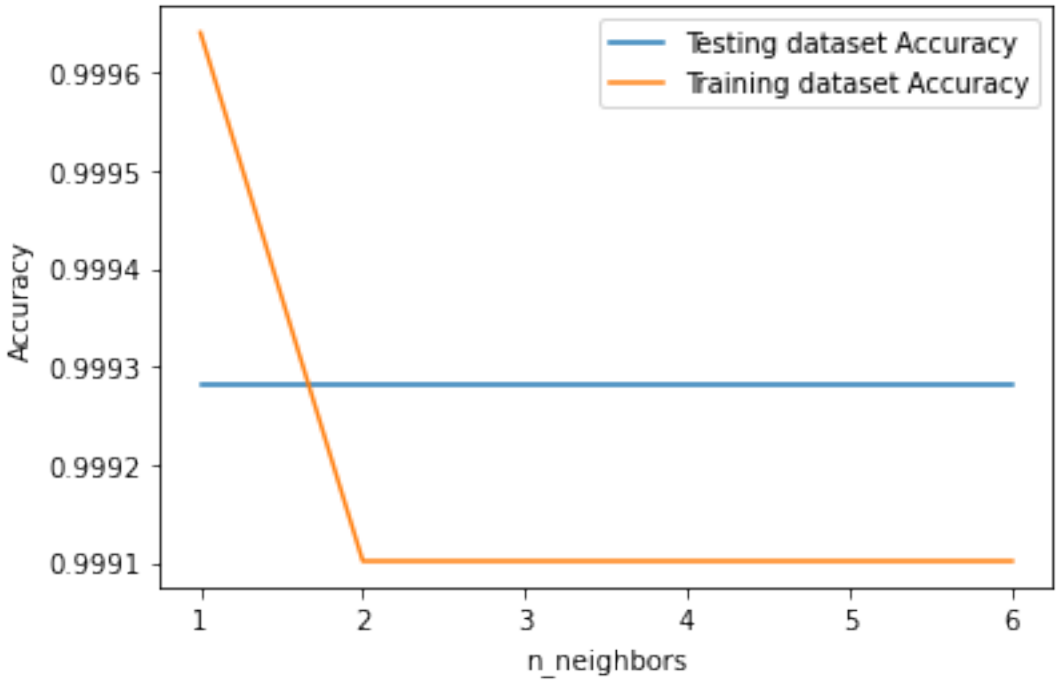
```
6
[ 556 1777 3075 4145 5269 6276]
```

[6]:
```
peaks = [0]*(len(temp[' frameRotationalSpeedZ ']))
for peak in tempy :
    peaks[peak] = 1
```

```
print(len(peaks))
print(len(temp))
temp["IsPeak"] = peaks
y = temp["IsPeak"].values#Detecs peaks
```

```
6956
6956
```

[7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

neighbors = np.arange(1, 7)#create an array from 1 to 199

train_accuracy = np.empty(len(neighbors))#both are random value arrays
test_accuracy = np.empty(len(neighbors))#with a length of 200


for i, k in enumerate(neighbors):# is the index and k the value of the array␣
↪neighbors
    knn = KNeighborsClassifier(n_neighbors=k)
    #aqui va bien
    knn.fit(X_train, y_train)

    train_accuracy[i] = knn.score(X_train, y_train)
    test_accuracy[i] = knn.score(X_test, y_test)

plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')


plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
```



[8]:
```
# Linear Reg trial
```

[10]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

reg = LogisticRegression()
```

[11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[12]:
```
reg.fit(X_train, y_train)
```

[12]:
```
LogisticRegression()
```

[13]:
```
pred = reg.predict(X_test)
print(pred)
```

```
[0 0 0 … 0 0 0]
```

[14]:
```
reg.score(X_test, y_test)
```

[14]:
```
0.9992816091954023
```

[15]:
```
reg.score(pred.reshape(-1,1),y_test)#maybe remove it
```

```
/opt/jupyterhub/anaconda/lib/python3.8/site-packages/sklearn/base.py:445:
UserWarning: X does not have valid feature names, but LogisticRegression was
fitted with feature names
  warnings.warn(
```

[15]:
```
0.9992816091954023
```

[ ]:
```

```