Machine Learning Assignment

Logistic Regression

Analysis

- In this Assignment we used dataset of League of Legends to predict winner based on Tower Kills, Inhibitor Kills, Dragon Kills, Baron Kills:

Import

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

Dataset

In [2]:

```
data = pd.read_csv('games.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	gameId	creationTime	gameDuration	seasonId	winner	firstBlood	firstTower	firstInhil
0	3326086514	1504279457970	1949	9	1	2	1	
1	3229566029	1497848803862	1851	9	1	1	1	
2	3327363504	1504360103310	1493	9	1	2	1	
3	3326856598	1504348503996	1758	9	1	1	1	
4	3330080762	1504554410899	2094	9	1	2	1	

5 rows × 61 columns

Since in our data winner 1 is for team 1 and 2 is for team 2. So we will make it to 0 and 1 for logistic regression.

In [4]:

```
data['winner'] = data['winner'].replace(1, 0)
data['winner'] = data['winner'].replace(2, 1)
data['winner'].unique()
```

Out[4]:

```
array([0, 1], dtype=int64)
```

In [5]:

```
data
```

Out[5]:

	gameId	creationTime	gameDuration	seasonId	winner	firstBlood	firstTower	firstInhibitor	first
0	3326086514	1504279457970	1949	9	0	2	1	1	
1	3229566029	1497848803862	1851	9	0	1	1	1	
2	3327363504	1504360103310	1493	9	0	2	1	1	
3	3326856598	1504348503996	1758	9	0	1	1	1	
4	3330080762	1504554410899	2094	9	0	2	1	1	
...	...	...	...	...	...	...	...	...	
51485	3308904636	1503076540231	1944	9	1	1	2	2	
51486	3215685759	1496957179355	3304	9	1	1	1	2	
51487	3322765040	1504029863961	2156	9	1	2	2	2	

Why features selection?

- As we know in LoL Towerkill and Inhibitorkill can best predict which team is doing well then Baronkill and Dragonkill
- So we will analyze how different features give us prediction

In [6]:

```
X = data[['t1_baronKills', 't1_dragonKills', 't2_baronKills', 't2_dragonKills']]
y = data['winner']
```

In [7]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=101)
logmodel = LogisticRegression()
logmodel.fit(X_train, y_train)
predictions = logmodel.predict(X_test)
```

In [8]:

```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.80      0.88      0.84      7733
           1       0.87      0.78      0.82      7714

    accuracy                           0.83     15447
   macro avg       0.83      0.83      0.83     15447
weighted avg       0.83      0.83      0.83     15447
```

In [9]:

```
X = data[['t1_towerKills','t1_inhibitorKills','t2_towerKills','t2_inhibitorKills']]
y = data['winner']
```

In [10]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=101)
logmodel = LogisticRegression()
logmodel.fit(X_train, y_train)
predictions = logmodel.predict(X_test)
```

In [11]:

```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      8484
           1       0.97      0.95      0.96      8508

    accuracy                           0.96     16992
   macro avg       0.96      0.96      0.96     16992
weighted avg       0.96      0.96      0.96     16992
```

Combining

- We saw we got accuracy of 83% for Baronkill and Dragonkill
- We got accuracy og 96% forTowerkill and Inhibitorkill
- We got accuracy of 96% by combining them. As our system already performing well of 96% so not much change by adding baron and dragon kill

In [12]:

```
X = data[['t1_baronKills', 't1_dragonKills', 't2_baronKills', 't2_dragonKills',
         't1_towerKills','t1_inhibitorKills','t2_towerKills','t2_inhibitorKills']]
y = data['winner']
```

In [13]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=101)
logmodel = LogisticRegression()
logmodel.fit(X_train, y_train)
predictions = logmodel.predict(X_test)
```

In [14]:

```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      8484
           1       0.97      0.95      0.96      8508

    accuracy                           0.96     16992
   macro avg       0.96      0.96      0.96     16992
weighted avg       0.96      0.96      0.96     16992
```

PCA

- Making new data set for ease
- Reducing it to two dimension
- Since in our data values of features vary so we will import standardscalar

In [15]:

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

In [16]:

```
new_data = pd.DataFrame(data[['winner', 't1_baronKills', 't1_dragonKills', 't2_baronKills',
         't1_towerKills','t1_inhibitorKills','t2_towerKills','t2_inhibitorKills']])
```

In [17]:

```
new_data
```

Out[17]:

	winner	t1_baronKills	t1_dragonKills	t2_baronKills	t2_dragonKills	t1_towerKills	t1_inhi
0	0	2	3	0	1	11	
1	0	0	2	0	0	10	
2	0	1	1	0	1	8	
3	0	1	2	0	0	9	
4	0	1	3	0	1	9	
...	...	...	...	...	...	...	
51485	1	0	0	0	4	2	
51486	1	0	2	4	4	5	
51487	1	0	1	0	2	0	
51488	1	0	0	0	1	0	
51489	0	1	2	0	1	11	

51490 rows × 9 columns

In [18]:

```
scaler=StandardScaler()
scaler.fit(new_data)
scaled_data=scaler.transform(new_data)
```

In [19]:

```
scaled_data[0]
```

Out[19]:

```
array([-0.98718638,  2.78752472,  1.33643434, -0.67541854, -0.33023792,
        1.39498948, -0.01387537, -0.14231362, -0.78413297])
```

In [20]:

```
pca=PCA(n_components=2)
pca.fit(scaled_data)
x_pca=pca.transform(scaled_data)
```

In [21]:

```
scaled_data.shape
```

Out[21]:

```
(51490, 9)
```

In [22]:

```
x_pca.shape
```

Out[22]:

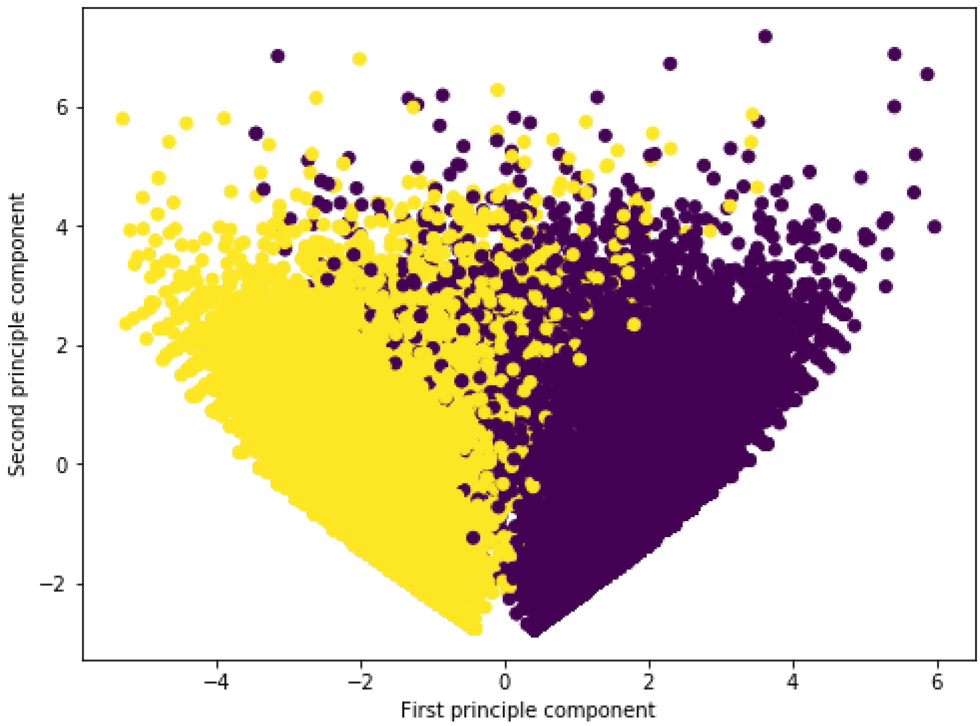
```
(51490, 2)
```

In [23]:

```
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],c=new_data['winner'])
plt.xlabel('First principle component')
plt.ylabel('Second principle component')
```

Out[23]:

```
Text(0, 0.5, 'Second principle component')
```



Applying Logistic Regression

- We saw that we now got accuracy of 98%.

In [24]:

```
X_train_pca, X_test_pca, y_train, y_test = train_test_split(x_pca, y, test_size=0.33, rando

logmodel = LogisticRegression()
logmodel.fit(X_train_pca, y_train)
predictions = logmodel.predict(X_test_pca)
```

In [25]:

```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      8484
           1       0.98      0.98      0.98      8508

    accuracy                           0.98     16992
   macro avg       0.98      0.98      0.98     16992
weighted avg       0.98      0.98      0.98     16992
```

End