PREDICTING FOOTBALL WINS FOR 2003 TO 2009 FOR TENNESSEE STATE UNIVERSITY USING PYTHON AND DATA FROM THE SCHOOL WEBSITES

by Erika Harrell

Disclaimer: All opinions and conclusions in this project are those of the author and not of the U.S. Department of Justice.

Executive Summary

This project was conducted to predict whether Revenue Sport Universities (FBS) would win a football game and what factors are involved in winning a football game. Data on scores and other football statistics were collected from the school websites of all Revenue Sport Universities (FBS) for years 2003 to 2009. Data on the NFL games and NFL draft data was also collected. The data scraped included the end zones dive and plays, end zones pass and plays, field goal and plays, fumbles, interceptions, kick off and returns, punts and returns, penalties with and yards representing a single game after data wrangling and cleaning were completed. Machine learning was used to predict the winner of a football game using the data collected. Machine learning algorithms included in the machine learning library of Sklearn and Keras were used. The algorithms included Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, and Neural Network. The data was split into 70% of each school different year of model data and 30% of each school different year of model data to predict winner 70% and to test a football game. The model with the highest accuracy was the Neural Network with an accuracy of 0.8 and a precision of 0.8 and a recall of 0.8. The model with the lowest accuracy was the Decision Tree with an accuracy of 0.7 and a precision of 0.7 and a recall of 0.7. The model with the lowest precision was the Support Vector Machine with a precision of 0.6 and a recall of 0.8. The model with the lowest recall was the Logistic Regression with a recall of 0.6 and a precision of 0.8.

Web Scraping: getting data on location of games and scores from TSU websites

Data Cleaning: correcting names and deleting extra text and spaces from lists

Data Wrangling: putting data from TSU websites into lists (one list per year)

```
[nbr] !chmod +x /usr/local/bin/chromedriver
[nbr] !ls -la /usr/local/bin/chromedriver
[nbr] !chromedriver --version
[nbr] !pip install -q selenium webdriver-manager
[nbr] from selenium import webdriver
[nbr] from selenium.webdriver.chrome.service import Service
[nbr] from selenium.webdriver.chrome.options import Options
[nbr] from selenium.webdriver.common.by import By
[nbr] from selenium.webdriver.support.ui import WebDriverWait
[nbr] from selenium.webdriver.support.expected_conditions as EC
[nbr] from bs4 import BeautifulSoup
[nbr] from datetime import datetime, date, timedelta
[nbr] import time, re, pandas as pd, numpy as np
[nbr] from google.colab.drive import drive
[nbr] drive.mount('/content/drive')
[nbr] !cp -r /content/drive/MyDrive/Colab\ Notebooks/Project\ 2/chromedriver_linux64.zip .
[nbr] !unzip -o chromedriver_linux64.zip
[nbr] !mv chromedriver /usr/local/bin/chromedriver
```

```
# get number of students and CGPA scale
SELECT COUNT(*), MAX(cgpa)
FROM students;
```

```
COUNT(*)|MAX(cgpa)
1000|10.0
```

```
# display first and last students
SELECT * FROM students ORDER BY id ASC LIMIT 1;
SELECT * FROM students ORDER BY id DESC LIMIT 1;
```

```
1|Ray Ed Jones|Male|Computer Science|Ray Building 2|2001-11-11|Ray Ed Jones|Ray Ed Jones|Ray Ed Jones
```

```
select count(*) as [count] from [Carts_items]
where [Carts_id] = 1
order by [count] desc
[count]
-------
2
2
2
2
2
2
2
2
2
2
Table, Query, Server execution:
CPU = 0,000ms,
elapsed = 0,000ms.
```

```
select top(1)
count = (select count(*) from [Carts_items] as [t] where [t].[Carts_id] = [t0].[Carts_id]),
[t0].[Carts_id] as [Carts_id], [t0].[Date_of_creation] as [Date_of_creation],
[t0].[Id] as [Id], [t0].[Is_ordered] as [Is_ordered],
[t0].[Is_paid] as [Is_paid], [t0].[Is_shipped] as [Is_shipped],
[t0].[Is_taken] as [Is_taken], [t0].[Is_ready] as [Is_ready],
[t0].[Is_cancelled] as [Is_cancelled]
from [Carts] as [t0]
order by [count] desc
Table 'Carts'. Scan count 1, logical reads 2, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
```

```
[2:27:00]
[2:27:00] [Server] Client "Pikachu" connected to "Pikachu's Pokémon Center" [127.0.0.1:7777] (v0.0.0.0)
[2:27:00] [Server] Client "Pikachu" [127.0.0.1] is an Observer
[2:27:00] [Server] Client "Pikachu" [127.0.0.1] joined the game
[2:27:00]
[2:27:00] [Server] Pikachu: [0, 0, 0, 
```

```
SELECT DISTINCT
FROM [dbo].[Sales].[Customer] AS [Customer]
WHERE [Customer].[Name] = N'Bike World' AND [Customer].[PersonID] IS NULL
ORDER BY [Customer].[CustomerID] ASC
OFFSET 0 ROWS
FETCH NEXT 1000 ROWS ONLY
```

```
SELECT [t0].[CustomerID], [t0].[AccountNumber], [t0].[CustomerType], [t0].[rowguid], [t0].[ModifiedDate]
FROM [dbo].[Sales].[Customer] AS [t0]
WHERE ([t0].[Name] = @p0) AND ([t0].[PersonID] IS NULL)
ORDER BY [t0].[CustomerID]
OFFSET 0 ROWS
FETCH NEXT 1 ROWS ONLY
-- @p0: Input Nvarchar (4000) [Bike World]
```

CustomerID	AccountNumber	CustomerType	rowguid	ModifiedDate
29483	AW00029483	S	4a0f0a0d-0a0a-4a0a-8a0a-0a0a0a0a0a0a	2011-05-24 00:00:00

	PPG	assists	fieldgoals	fieldgoalsattempted	freethrows	freethrowsattempted	threepointers	threepointersattempted	rebounds	offensivereb	defensivereb	steals	blocks	turnovers	fouls
count	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
mean	19.4	4.1	7.1	15.7	4.8	6.3	1.4	4.1	10.4	2.3	8.1	1.3	0.8	2.1	2.1
std	6.1	2.1	2.2	3.4	2.1	2.5	1.1	2.1	3.2	1.5	2.8	1.1	0.9	1.4	1.4
min	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	15.0	2.0	5.0	13.0	3.0	4.0	0.0	2.0	8.0	1.0	6.0	0.0	0.0	1.0	1.0
50%	19.0	4.0	7.0	16.0	4.0	6.0	1.0	4.0	10.0	2.0	8.0	1.0	1.0	2.0	2.0
75%	23.0	5.0	8.0	18.0	6.0	8.0	2.0	6.0	12.0	3.0	10.0	2.0	1.0	3.0	3.0
max	41.0	14.0	16.0	29.0	14.0	17.0	7.0	14.0	22.0	10.0	19.0	7.0	5.0	9.0	9.0

```
[1 rows x 15 columns]
```

```
# Merge data frames with TSU scores and annual stats
df = pd.merge(df2,df1,left_on='year',right_index=True)
df.shape
```

```
(1000, 24)
```

```
[2]: ‘Year’ = df.groupby(‘Year’).sum()[‘Year’].sort_values(ascending=False).head(10).reset_index()
Year = df.groupby(‘Year’).sum()[‘Year’].sort_values(ascending=False).head(10).reset_index()
plt.title('Year Difference by Year')
sns.barplot(x='Year', y='Year', data=Year)
plt.xticks(rotation=90)
plt.show()
```

```
[2]: ‘Year’ = df.groupby(‘Year’).sum()[‘Year’].sort_values(ascending=False).head(10).reset_index()
Year = df.groupby(‘Year’).sum()[‘Year’].sort_values(ascending=False).head(10).reset_index()
plt.title('Year Difference by Year')
sns.barplot(x='Year', y='Year', data=Year)
plt.xticks(rotation=90)
plt.show()
```

```
[2]: ‘Year’ = df.groupby(‘Year’).sum()[‘Year’].sort_values(ascending=False).head(10).reset_index()
Year = df.groupby(‘Year’).sum()[‘Year’].sort_values(ascending=False).head(10).reset_index()
plt.title('Year Difference by Year')
sns.barplot(x='Year', y='Year', data=Year)
plt.xticks(rotation=90)
plt.show
```

```
[1]: date  city  season  Tmax
     <dbl> <chr> <chr>   <dbl>
  1. 19140101 Dallas  Winter 50.1
  2. 19140102 Dallas  Winter 49.9
  3. 19140103 Dallas  Winter 62.1
  4. 19140104 Dallas  Winter 65.9
  5. 19140105 Dallas  Winter 62.1
  6. 19140106 Dallas  Winter 55.9
  7. 19140107 Dallas  Winter 59.0
  8. 19140108 Dallas  Winter 62.1
```

```
[2]: ggplot(data = temp, aes(x = date, y = Tmax, color = season)) +
     geom_point(size = 0.5) +
     ggtitle("Tmax by Year")
```

```
[3]: ggplot(data = temp, aes(x = Tmax, fill = season)) +
     geom_density(alpha = 0.5) +
     ggtitle("Tmax by Year")
```

```
[4]: ggplot(data = temp, aes(x = Tmax, y = season, fill = season)) +
     geom_boxplot(alpha = 0.5) +
     ggtitle("Tmax by Year")
```

```
[5]: ggplot(data = temp, aes(x = Tmax, y = season, fill = season)) +
     geom_violin(alpha = 0.5) +
     ggtitle("Tmax by Year")
```

```
[6]: ggplot(data = temp, aes(x = Tmax, y = season, fill = season)) +
     geom_ridgeline(alpha = 0.5) +
     ggtitle("Tmax by Year")
```

```
[7]: ggplot(data = temp, aes(x = Tmax, y = season, fill = season)) +
     geom_ridgeline(alpha = 0.5) +
     ggtitle("Tmax by Year")
```

```
[8]: ggplot(data = temp, aes(x = Tmax, y = season, fill = season)) +
     geom_ridgeline(alpha = 0.5) +
     ggtitle("Tmax by Year")
```

```
```