MECE 6397: Project 4

Syam Evani
08/09/2025

Introduction

Table of Contents

My code for this project is included in three ways:

- Embedded in portions with relevant discussion
- Uploaded as part of this deliverable on Canvas
- Available on a public github here: https://github.com/Sam-v6/mece-6397-doe/tree/main/project4

Part 1: Dataset Selection

As an overly zealous NFL football fan, I often wonder about the statistical significance of various betting odds as predictors for what team will win the game. Thus I was interested in finding some accessible data (not getting too involved in API data calls and such but a simple csv) that contains NFL game outcomes and various betting data. I was able to locate a great dataset that included a plethora of data. This dataset summarizes every NFL game since 1999 to present day including the outcome, total points scored, various betting odds, where the game was played, what day it was played, whether it was a regular season or playoff game, along with a wealth of other data even diving into the playing surface, weather, etc. My ultimate goal with this data was to take some betting data and a few other factors to see how statistically significant they are in the home team winning the game.

Dataset link: https://github.com/nflverse/nfldata/blob/master/data/games.csv .

Part 2: Dataset preparation

To prepare the data I load it in and immediately drop several columns that I'm not interested in. These include things like game IDs, stadiums, coaches, players, etc. I also then cut out some games that haven't been played yet (2024 season) and eliminate any neutral site games as I want to evalute games where there is a clear away and home team.

I select several features to evaluate including:

- Type of game (regular season, playoff, conference championship)
- Day of week game is played
- Days away team has had to rest
- Days home team has had to rest
- Away moneyline
- Home moneyline
- Overall spread
- Away spread odds
- Home spread odds
- Whether it is divisional game (are teams in the same division)
- Playing surface
- Temperature
- Wind Speed

I then go through the dataset and make sure I'm only evaluating data that has all of these features for every game. Some games since 1999 don't have all this recorded, hence, making this step neeeded.

At this point I can categorize the data, shifting game type, weekday, and surface from their string formats to enumerations that I can actually process downstream. Finally I evalute the final score to determine if the home team lost (0) or won (1).

In [ ]:
```
"""
Purpose: Project 4 - ML analysis on nfl betting info to determine wins
Author: Syam Evani
"""

# Standard imports
import os

# Additional imports
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from scipy.stats import f
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
```

```
from scipy.stats import t, ttest_ind

# Local imports
# None

#----------------------------------------------------
# Load data
#----------------------------------------------------
data = pd.read_csv(os.path.join(os.getenv('USERPROFILE'),'repos','mece-6397-doe','p

# There's alot of superflous data we don't want to draw predictions from, let's dro
drop_list = ["week", "gameday", "gametime", "old_game_id", "gsis", "nfl_detail_id",
             "pfr", "pff", "espn", "ftn", "away_qb_id", "home_qb_id", "away_qb_name
             "home_qb_name" , "away_coach", "home_coach", "referee", "stadium_id",
             "overtime", "season", "away_team", "home_team", "roof"]
data = data.drop(columns=drop_list)

# This dataset includes games that haven't been played yet, filter out 2024 games a
data = data[~data['game_id'].str.contains("2024", na=False)]
data = data[~data['location'].str.contains("Neutral", na=False)]
data = data.drop(columns=['game_id'])
data = data.drop(columns=['location'])

# Feature selection
selected_features = ['game_type', 'weekday', 'away_rest',
                     'home_rest', 'away_moneyline','home_moneyline',
                     'spread_line', 'away_spread_odds', "home_spread_odds",
                     "div_game", "surface", "temp", "wind"]

# Drop rows with missing data in any of the selected features columns
data = data.dropna(subset=selected_features)

# Check on our data and see what factors we are considering now
print(data.columns)

#----------------------------------------------------
# Categorize data
#----------------------------------------------------
# Clean up data to go from strings to enumerations
cleanup_nums = {
                "game_type": {"REG": 1, "WC": 2, "DIV":3, "CON":4, "SB":5},
                "weekday": {"Monday": 1, "Tuesday":2, "Wednesday":3, "Thursday":4,
                "surface": {"sportturf":0, 'astroplay':0, 'grass':1, 'fieldturf':0,
                }
data = data.replace(cleanup_nums)

# Create a new column based on the 'result' column
data['home_team_win'] = np.where(data['result'] > 0, 1, 0)

# Print data
print(data.head())
print(data.tail())
```

```
Index(['game_type', 'weekday', 'away_score', 'home_score', 'result', 'total',
       'away_rest', 'home_rest', 'away_moneyline', 'home_moneyline',
       'spread_line', 'away_spread_odds', 'home_spread_odds', 'total_line',
       'under_odds', 'over_odds', 'div_game', 'surface', 'temp', 'wind'],
      dtype='object')
      game_type  weekday  away_score  home_score  result  total  away_rest  \
1892          1        7        13.0        16.0     3.0   29.0         14
1893          1        7        12.0        17.0     5.0   29.0          7
1894          1        7        18.0        21.0     3.0   39.0          6
1896          1        7         0.0        41.0    41.0   41.0          7
1897          1        7        31.0        28.0    -3.0   59.0          7

      home_rest  away_moneyline  home_moneyline  ...  away_spread_odds  \
1892          7          -107.0          -103.0  ...            -112.0
1893          7           119.0          -129.0  ...             113.0
1894          7           300.0          -330.0  ...             106.0
1896         14           293.0          -323.0  ...            -103.0
1897          7          -369.0           339.0  ...             100.0

      home_spread_odds  total_line  under_odds  over_odds  div_game  surface  \
1892             104.0        33.5      -103.0     -107.0         0        0
1893            -121.0        34.5      -106.0     -104.0         0        0
1894            -114.0        41.0      -102.0     -108.0         1        1
1896            -105.0        39.0       100.0     -110.0         0        1
1897            -108.0        46.5      -106.0     -104.0         0        0

      temp  wind  home_team_win
1892  68.0   7.0              1
1893  58.0  11.0              1
1894  82.0   6.0              1
1896  82.0  15.0              1
1897  62.0   2.0              0

[5 rows x 21 columns]
      game_type  weekday  away_score  home_score  result  total  away_rest  \
6699          3        6        10.0        34.0    24.0   44.0          7
6700          3        6        21.0        24.0     3.0   45.0          6
6702          3        7        27.0        24.0    -3.0   51.0          8
6703          4        7        17.0        10.0    -7.0   27.0          7
6704          4        7        31.0        34.0     3.0   65.0          7

      home_rest  away_moneyline  home_moneyline  ...  away_spread_odds  \
6699         14           330.0          -425.0  ...            -115.0
6700         13           360.0          -470.0  ...            -112.0
6702          6           120.0          -142.0  ...            -110.0
6703          8           180.0          -218.0  ...            -110.0
6704          8           260.0          -325.0  ...            -120.0

      home_spread_odds  total_line  under_odds  over_odds  div_game  surface  \
6699            -105.0        44.0      -105.0     -115.0         0        1
6700            -108.0        50.5      -105.0     -115.0         0        1
6702            -110.0        45.5      -108.0     -112.0         0        0
6703            -112.0        44.0      -105.0     -115.0         0        1
6704             100.0        52.5      -110.0     -110.0         0        1

      temp  wind  home_team_win
```

```
6699  27.0  16.0             1
6700  59.0   8.0             1
6702  25.0  11.0             0
6703  47.0   7.0             0
6704  69.0   5.0             1

[5 rows x 21 columns]
```

Part 3: T-testing

I slice my data into the usual 80% for training and 20% for testing with our selected features with the home team victory status as the target.

In the same fashion as the sample code I perform a T-testing with one minor modification to record whether or not the specific feature is significant in impacting the outcome for the home team (based on if the p value is below 0.05 and if the absolute value of the t value is larger than the critical t value)

Results are shown below:

Feature	Significant?	T-statistic	P-value	Critical
surface	SIGNIFICANT	-3.7055993914058156	0.0002144236330758636	-1.6453250
wind	Not important	-0.024027255376345074	0.9808323511483465	-1.6453250
game_type	Not important	1.7626017611319151	0.07806215295834043	-1.6453250
weekday	Not important	-0.15812049374843445	0.8743717271701335	-1.6453250
away_rest	SIGNIFICANT	-2.3750130051411737	0.017606345955603387	-1.6453250
home_rest	Not important	0.974192675505116	0.33003376365095805	-1.6453250
away_moneyline	SIGNIFICANT	21.884464420578997	4.078382882439177e-99	-1.6453250
home_moneyline	SIGNIFICANT	-20.612840525524433	8.492000533979854e-89	-1.6453250
spread_line	SIGNIFICANT	23.14438175202809	8.727713364946639e-110	-1.6453250
away_spread_odds	SIGNIFICANT	-6.146854246433647	8.869240168908078e-10	-1.6453250
home_spread_odds	SIGNIFICANT	3.9606055931091095	7.637211505173663e-05	-1.6453250
div_game	SIGNIFICANT	-2.6049679456009094	0.009230384202277137	-1.6453250
temp	Not important	-1.7575713625833678	0.07891510083042995	-1.6453250

Not suprising (at least for myself as fan of typically bad NFL teams and used to seeing dreadful predicted spreads for my team to lose), betting stats have a strong signficance in prediciting the outcome of the game. Interstingly, the surface of the playing field is rather important, indicating for example the home team's winning probabilty is signifcantly lower on turf compared to playing on grass.

In [ ]:
```
#-----------------------------------------------------
# Slice data into training and testing sets
#-----------------------------------------------------
# Split training data
X = data[selected_features]
y = data['home_team_win']

# Create a StandardScaler object
scaler = StandardScaler()

# Fit and transform the data
X_normalized = scaler.fit_transform(X)

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_normalized, y, test_size=0.2,

#-----------------------------------------------------
# T-test
#-----------------------------------------------------
t_results = {}
print("-------------------------------------------")
print("T Results")
print("-------------------------------------------")
for feature in selected_features:
    win = data[data['home_team_win'] == 1][feature]
    loss = data[data['home_team_win'] == 0][feature]

    # Calculate t-test statistic and p-value
    t_stat, p_value = ttest_ind(win, loss)

    # Calculate critical t-value from t-distribution
    n1 = len(win)
    n2 = len(loss)
    dof = n1 + n2 - 2  # Degrees of freedom for independent two-sample t-test
    critical_t = t.ppf(0.05, dof)  # Using 0.05 significance level

    if p_value < 0.05 and abs(t_stat) > critical_t:
        print(f"SIGNIFICANT: T-test results for '{feature}': t-statistic={t_stat},
    else:
        print(f"Not important: T-test results for '{feature}': t-statistic={t_stat}
```

```
----------------------------------------
T Results
----------------------------------------
Not important: T-test results for 'game_type': t-statistic=1.7626017611319151, p-val
ue=0.07806215295834043, critical t-value=-1.6453250802533461
Not important: T-test results for 'weekday': t-statistic=-0.15812049374843445, p-val
ue=0.8743717271701335, critical t-value=-1.6453250802533461
SIGNIFICANT: T-test results for 'away_rest': t-statistic=-2.3750130051411737, p-valu
e=0.017606345955603387, critical t-value=-1.6453250802533461
Not important: T-test results for 'home_rest': t-statistic=0.974192675505116, p-valu
e=0.33003376365095805, critical t-value=-1.6453250802533461
SIGNIFICANT: T-test results for 'away_moneyline': t-statistic=21.884464420578997, p-
value=4.078382882439177e-99, critical t-value=-1.6453250802533461
SIGNIFICANT: T-test results for 'home_moneyline': t-statistic=-20.612840525524433, p
-value=8.492000533979854e-89, critical t-value=-1.6453250802533461
SIGNIFICANT: T-test results for 'spread_line': t-statistic=23.14438175202809, p-valu
e=8.727713364946639e-110, critical t-value=-1.6453250802533461
SIGNIFICANT: T-test results for 'away_spread_odds': t-statistic=-6.146854246433647,
p-value=8.869240168908078e-10, critical t-value=-1.6453250802533461
SIGNIFICANT: T-test results for 'home_spread_odds': t-statistic=3.9606055931091095,
p-value=7.637211505173663e-05, critical t-value=-1.6453250802533461
SIGNIFICANT: T-test results for 'div_game': t-statistic=-2.6049679456009094, p-value
=0.009230384202277137, critical t-value=-1.6453250802533461
SIGNIFICANT: T-test results for 'surface': t-statistic=-3.7055993914058156, p-value=
0.0002144236330758636, critical t-value=-1.6453250802533461
Not important: T-test results for 'temp': t-statistic=-1.7575713625833678, p-value=
0.07891510083042995, critical t-value=-1.6453250802533461
Not important: T-test results for 'wind': t-statistic=-0.024027255376345074, p-value
=0.9808323511483465, critical t-value=-1.6453250802533461
```

Part 4: Model creation

Drawing on our first homework assignment, I used multiple regression models including linear, decision tree, random forest, SVR, and KNN to build models and evalute how a collection of all of our features (even those that were not significant for the first process of building the model) could perform. Results are shown below. For this specific collection of features the best model would actually be simple linear regression.

Model	Value
lr	0.21056065180003614
dtr	0.4188562596599691
rfr	0.2356763523956723
svr	0.25144313085577946
knn	0.24500772797527048

In [ ]:
```
#-----------------------------------------------------
# Model developement
#-----------------------------------------------------
models = {}
results = {}

# Train different models
models["lr"] = LinearRegression().fit(X_train, y_train)
models["dtr"] = DecisionTreeRegressor().fit(X_train, y_train)
models["rfr"] = RandomForestRegressor().fit(X_train, y_train)
models["svr"] = SVR().fit(X_train, y_train)                          # Default rbf k
models["knn"]  = KNeighborsRegressor().fit(X_train, y_train)         # By default wi

# Predict and calculate MSE
for regressor in models:
    y_pred = models[regressor].predict(X_test)
    mse = mean_squared_error(y_test, y_pred)

    # Store the results
    results[regressor] = mse

print("-------------------------------------------")
print("Model Results")
print("-------------------------------------------")
print(results)
```

```
-------------------------------------------
Model Results
-------------------------------------------
{'lr': 0.21056065180003614, 'dtr': 0.4188562596599691, 'rfr': 0.2356763523956723, 's
vr': 0.25144313085577946, 'knn': 0.24500772797527048}
```