National_leagues_prediction

December 7, 2022

1 1. Imports

[ ]:
```
import pandas as pd
import numpy as np
import helper

%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[ ]:
```
pd.set_option("display.max_columns", 30)
```

[ ]:
```
data_players_raw = pd.read_csv(f"Fifa_players/FIFA22_official_data.csv")
```

2 2. Get data on fifa players

[ ]:
```
def get_clean_fifa_ratings(year):
    # Read CSV file of a specific year, clean the data and store it in a␣
 ↪DataFrame
    data_fifa_players = pd.read_csv(f"Fifa_players/FIFA{year}_official_data.
 ↪csv")
    data_fifa_players = data_fifa_players.set_index('ID')
    data_fifa_players = data_fifa_players[data_fifa_players["Overall"] >= 55]

    # Clean players' position
    data_fifa_players["Position"] = data_fifa_players["Position"].str.
 ↪extract(r">(\w+)")
    if "Best Position" in data_fifa_players.columns:
        data_fifa_players['Position'] = data_fifa_players['Best Position'].
 ↪fillna(value=data_fifa_players['Position'])

    # Keep only desired columns
    data_fifa_players = data_fifa_players[['Name', 'Nationality', 'Overall',␣
 ↪'Position', 'Club']].dropna()
```

```
    data_fifa_players["Year"] = [year]*len(data_fifa_players)
    data_fifa_players['Club'] = data_fifa_players['Club'].str.upper()

    # Remove old players that have a name starting with a number (e.g. "14 D.␣
↪Beckam")
    data_fifa_players['Name'] = data_fifa_players['Name'].transform(lambda x:␣
↪np.NaN if (x.lstrip(" ")[0] == "0" or x.lstrip(" ")[0] == "1" or x.lstrip("␣
↪")[0] == "2") else x)
    data_fifa_players = data_fifa_players.dropna()

    # We could improve the Overall (rating) with the potential of the player,␣
↪the age, etc...

    return data_fifa_players

def clean_position(x, data_previous_year):
    # Function applied to a dataframe in the function "get_all_fifa_ratings"
    # Tries to reduce the number of players that have a postion names:
    # SUB for substitute or RES for reserviste (in which case we don't know␣
↪there
    # position on the field!)
    if x["Position"] not in ["RES", "SUB"]:
        return x
    else:
        try:
            x["Position"] = data_previous_year.loc[x.name]["Position"]
        except KeyError:
            pass
    return x

def get_all_fifa_ratings():
    # Put all the data together (for each of the years) and clean the players'␣
↪position
    data_all = pd.DataFrame()
    for year in range(17, 24):
        data_year = get_clean_fifa_ratings(year)
        if year >= 18:
            data_year = data_year.apply(clean_position,␣
↪args=[data_all[data_all["Year"] == (year-1)]], axis=1)
        data_all = pd.concat([data_all, data_year])
    return data_all
```

[ ]:
```
data_all_players = get_all_fifa_ratings()
```

3 3. Get data on games

[ ]:
```python
def match_club_names(x):
    # to be applied to a DataFrame
    # Changes the names of the club to match those in fifa
    if x["home_team"] in helper.DICT_TO_CLUB_NAME_DATA_TO_FIFA.keys():
        x["home_team"] = helper.DICT_TO_CLUB_NAME_DATA_TO_FIFA[x["home_team"]]
    if x["away_team"] in helper.DICT_TO_CLUB_NAME_DATA_TO_FIFA.keys():
        x["away_team"] = helper.DICT_TO_CLUB_NAME_DATA_TO_FIFA[x["away_team"]]
    return x

def get_result(x):
    # to be applied to a DataFrame
    # to add a column mentionning the result of the game
    if x["home_score"] > x["away_score"]:
        return "w_home"
    elif x["home_score"] < x["away_score"]:
        return "w_away"
    return "draw"

def find_team_recent_shape(x, data_games, location, nb_old_games=3):
    # to be applied to a DataFrame
    # Get the result of the teams on the past previous games
    old_games_team = data_games[
        (data_games["home_team"] == x[f"{location}_team"]) +
        (data_games["away_team"] == x[f"{location}_team"])].loc[:x.name].iloc[:
 ↪-1]

    if len(old_games_team) < nb_old_games:
        return x
    else:
        for i in range(1, nb_old_games + 1):
            old_game = old_games_team.iloc[-i]
            if old_game["home_team"] == x[f"{location}_team"]:
                if old_game["Result"] == "w_home":
                    x[f"{location}_{i}_games_ago"] = "win"
                elif x["Result"] == "draw":
                    x[f"{location}_{i}_games_ago"] = "draw"
                else:
                    x[f"{location}_{i}_games_ago"] = "lose"
            else:
                if old_game["Result"] == "w_home":
                    x[f"{location}_{i}_games_ago"] = "lose"
                elif x["Result"] == "draw":
                    x[f"{location}_{i}_games_ago"] = "draw"
                else:
                    x[f"{location}_{i}_games_ago"] = "win"
```

```python
        return x

def alternative_find_team_recent_shape(x, data_games, location, nb_old_games=3):
    # to be applied to a DataFrame
    # Similar to the previous function but puts the goals difference of the past few games
    # instead of just the result
    old_games_team = data_games[
        (data_games["home_team"] == x[f"{location}_team"]) +
        (data_games["away_team"] == x[f"{location}_team"])].loc[:x.name].iloc[:-1]

    if len(old_games_team) < nb_old_games:
        return x
    else:
        for i in range(1, nb_old_games + 1):
            old_game = old_games_team.iloc[-i]
            if old_game["home_team"] == x[f"{location}_team"]:
                x[f"{location}_{i}_games_ago"] = old_game["home_score"] - old_game["away_score"]
            else:
                x[f"{location}_{i}_games_ago"] = old_game["away_score"] - old_game["home_score"]
        return x


def get_clean_games_leagues(list_leagues_to_get=["france", "england"], nb_old_games=3):
    # Gets the data from the games of various leagues
    folder_path = "data_leagues/results/"
    data_games = pd.DataFrame()
    for league in list_leagues_to_get:
        data_games = pd.concat([
            data_games,
            pd.read_csv(folder_path + league + ".csv", index_col='date', parse_dates=True).loc["2016-04":]],
            axis=0)

    data_games = data_games[["home", "away", "gh", "ga", "competition"]].rename(
        columns={
            "home": "home_team",
            "away": "away_team",
            "gh": "home_score",
            "ga": "away_score",
            "competition": "tournament"
        }
```

```
    ).reset_index()

    data_games['home_team'] = data_games['home_team'].str.upper()
    data_games['away_team'] = data_games['away_team'].str.upper()
    data_games = data_games.apply(match_club_names, axis=1)

    # Transform the score into win, lose or draw
    data_games["Result"] = data_games.apply(get_result, axis=1)

    # Add shape of the teams on their previous games
    for location in ["home", "away"]:
        for i in range(1, nb_old_games+1):
            data_games[f"{location}_{i}_games_ago"] = np.NaN
    data_games = data_games.apply(alternative_find_team_recent_shape,␣
↪args=(data_games, "home", nb_old_games), axis=1)
    data_games = data_games.apply(alternative_find_team_recent_shape,␣
↪args=(data_games, "away", nb_old_games), axis=1)

    data_games['Year'] = data_games["date"].transform(lambda x: int(x.year) -␣
↪2000 if x.month < 8 else int(x.year) + 1 - 2000)
    data_games = data_games[data_games['Year'] > 16]
    data_games = data_games.set_index('date')



    data_games["home_team_year"] = data_games["home_team"] + "_" +␣
↪data_games["Year"].astype(str)
    data_games["away_team_year"] = data_games["away_team"] + "_" +␣
↪data_games["Year"].astype(str)

    return data_games
```

[ ]:
```
data_games = get_clean_games_leagues(list_leagues_to_get=["france", "england",␣
↪"spain", "germany", "uefa-cl", "uefa-cw"], nb_old_games=5)
```

[ ]:
```
# old_games_team = data_games[
#     (data_games["home_team"] == "AS NANCY LORRAINE") +
#     (data_games["away_team"] == "AS NANCY LORRAINE")].loc[:"2017-04-08"].
↪iloc[:-1]
```

[ ]:
```
# data_all_players[data_all_players["Club"].str.contains("BOURNE")]
```

[ ]:
```
# TEST: Get all of the potential players in a team
# data_all_players[
#     (data_all_players["Club"] == "RACING CLUB DE LENS") &
#     (data_all_players["Year"] == 20)
```

```
# ].sort_values("Overall", ascending=False)
```

[ ]:
```
# TEST: Check which teams aren't in the fifa database (their names have
# to be matched with those in fifa)
# def manual_cleaning_names(data_games):

# list_teams_to_replace = []
# for team in data_games["home_team"].unique():
#     if team not in list(data_all_players["Club"]):
#         print(f"Name games : {team}")
#         list_teams_to_replace.append(team)
# print(list_teams_to_replace)
```

4 4. Put fifa players in their teams

[ ]:
```
def get_potential_players(players_country, list_fifa_positions):
    potential_players = pd.DataFrame(columns=["ID", "Name", "Overall"])
    for fifa_position in list_fifa_positions:
        players_country_and_position =␣
↪players_country[players_country["Position"] == fifa_position].
↪sort_values("Overall", ascending=False)
        if len(players_country_and_position) != 0:
            potential_players = pd.concat(
                [
                    potential_players,
                    pd.DataFrame({
                        "ID": [players_country_and_position.iloc[0].name],
                        "Name": [players_country_and_position.iloc[0]["Name"]],
                        "Overall": [players_country_and_position.
↪iloc[0]["Overall"]]
                    })
                ],
                axis=0)
    return potential_players


def get_players_team(team, data_players, year, version, display=False):
    # version: "Club" or "Nationality"
    players_country = data_players[
        (data_players[version] == team) &
        (data_players["Year"] == year)]
    dict_positions = {
        "ST1":["ST", "CF", "LS", "RS"],
        "ST2":["ST", "CF", "LF", "RF", "LS", "RS", "RW", "LW"],
        "CM1":["CM", "LCM", "RCM", "CAM", "RAM", "LAM"],
        "CM2":["CM", "LCM", "RCM", "CDM", "RDM", "LDM"],
```

```
        "LM": ["LW", "LM", 'LAM', 'CAM'],
        "RM": ["RM", "RW", "RAM", 'CAM'],
        "CB1":["CB", "LCB", "RCB"],
        "CB2":["CB", "LCB", "RCB"],
        "LB": ["LWB", "LB", "CB", "LCB"],
        "RB": ["RWB", "RB", "CB", "RCB"],
        "GK": ["GK"]
        }

    list_overall_players_in_team = []
    for defined_position, list_fifa_positions in dict_positions.items():
        potential_players = get_potential_players(players_country,␣
↪list_fifa_positions)
            # print(players_country_and_position.iloc[0])
            # print(potential_players)
        if len(potential_players) == 0:
            list_fifa_positions = ["SUB", "RES"]
            potential_players = get_potential_players(players_country,␣
↪list_fifa_positions)
        if len(potential_players) == 0:
            potential_players = pd.DataFrame({
                "ID": [0],
                "Name": ["Average"],
                "Overall": [int(np.mean(list_overall_players_in_team)*0.95)]
            })
        else:
            potential_players = potential_players.sort_values("Overall",␣
↪ascending=False)
            players_country = players_country.drop(potential_players.iloc[0].
↪ID, axis=0)
        list_overall_players_in_team.append(potential_players.
↪iloc[0]["Overall"])
        if display:
            print(f"{defined_position}: - {list_fifa_positions} -␣
↪{potential_players.iloc[0]['Name']} - {potential_players.
↪iloc[0]['Overall']}")

    players_in_team = pd.DataFrame(columns=["Year", "Team"]+list(dict_positions.
↪keys()))
    players_in_team.loc[len(players_in_team)] = [year, team] +␣
↪list_overall_players_in_team

    return players_in_team
```

[ ]:
```
# TEST: get players of a specific team
```

```
# get_players_team("FC BAYERN MÜNCHEN", data_all_players, year=23,␣
↪display=True, version="Club")
```

[ ]:
```
def get_roster(data_games, data_all_players, version):
    # get the roster of every team in the games data
    data_rosters = pd.DataFrame()
    teams_not_found = []
    for year in data_all_players["Year"].unique():
        for team in (pd.concat([data_games[data_games["Year"] ==␣
↪year]["home_team"], data_games[data_games["Year"] == year]["away_team"]])).
↪unique():
            try:
                data_rosters = pd.concat([data_rosters, get_players_team(team,␣
↪data_all_players, year=year, version=version)], axis=0)
            except ValueError:
                teams_not_found.append(f"{year}_{team}")
                print(year, team)
    data_rosters["Team_year"] = data_rosters["Team"] + "_" +␣
↪data_rosters["Year"].astype(str)
    # print(teams_not_found)
    return data_rosters
```

[ ]:
```
data_rosters = get_roster(data_games, data_all_players, version="Club")
```

```
17 QARABAG FK
17 APOEL NIKOSIA
17 DINAMO TBILISI
17 RED STAR BELGRADE
17 DUNDALK FC
17 FC KOEBENHAVN
17 STEAUA BUCURESTI
17 HAPOEL BEER SHEVA
17 RB SALZBURG
17 PAOK SALONIKI
17 CELTIC FC
17 DINAMO ZAGREB
17 AFC AJAX
17 PFC LUDOGORETS RAZGRAD
17 VIKTORIA PLZEN
17 AS ROMA
17 FK ROSTOV
17 FC BASEL
17 DINAMO KIEV
17 PSV EINDHOVEN
17 CSKA MOSKVA
17 BESIKTAS
17 SSC NAPOLI
```

17 ALASHKERT FC
17 VIKINGUR GOTU
17 HIBERNIANS FC
17 THE NEW SAINTS
17 LINFIELD FC
17 KF TREPCA89
17 FC INFONET TALLINN
17 EUROPA FC
17 FC SANTA COLOMA
17 SP LA FIORITA
17 HNK RIJEKA
17 PARTIZAN
17 SPARTAKS JURMALA
17 FC SHERIFF
17 ZRINJSKI MOSTAR
17 MALMOE FF
17 BATE BORISOV
17 FK ZALGIRIS
17 IFK MARIEHAMN
17 MSK ZILINA
17 FH HAFNARFJOERDUR
17 FC ASTANA
17 VARDAR SKOPJE
17 FC SAMTREDIA
17 FK BUDUCNOST PODGORICA
17 F91 DUDELANGE
17 KS PERPARIMI KUKES
17 NK MARIBOR
17 BUDAPEST HONVED
17 SLAVIA PRAHA
17 AEK ATHEN
17 FC VIITORUL CONSTANTA
17 ASTRA GIURGIU
17 SPARTA PRAHA
17 OLYMPIAKOS PIRAEUS
17 FK PARTIZANI
17 FK AS TRENCIN
17 FENERBAHCE
17 ISTANBUL BASAKSEHIR FK
18 FC SHERIFF
18 CSKA MOSKVA
18 APOEL NIKOSIA
18 BATE BORISOV
18 ISTANBUL BASAKSEHIR FK
18 FC KOEBENHAVN
18 VIKTORIA PLZEN
18 FH HAFNARFJOERDUR
18 PFC LUDOGORETS RAZGRAD

18 AFC AJAX
18 HNK RIJEKA
18 OLYMPIAKOS PIRAEUS
18 QARABAG FK
18 CELTIC FC
18 HAPOEL BEER SHEVA
18 SSC NAPOLI
18 FC ASTANA
18 NK MARIBOR
18 SLAVIA PRAHA
18 STEAUA BUCURESTI
18 AS ROMA
18 SPARTAK MOSKVA
18 BESIKTAS
18 FC BASEL
18 FC SANTA COLOMA
18 SP LA FIORITA
18 LINCOLN RED IMPS
18 TORPEDO KUTAISI
18 FC FLORA TALLINN
18 F91 DUDELANGE
18 ALASHKERT FC
18 VIKINGUR GOTU
18 KF SHKENDIJA 79
18 KF DRITA GJILAN
18 FK SUDUVA
18 SPARTAKS JURMALA
18 SPARTAK TRNAVA
18 KS PERPARIMI KUKES
18 OLIMPIJA LJUBLJANA
18 VALUR REYKJAVIK
18 VALLETTA FC
18 MALMOE FF
18 THE NEW SAINTS
18 MOL VIDI FC
18 RED STAR BELGRADE
18 CRUSADERS FC
18 ZRINJSKI MOSTAR
18 FK SUTJESKA
18 CFR CLUJ
18 PAOK SALONIKI
18 DINAMO ZAGREB
18 AEK ATHEN
18 FC VIITORUL CONSTANTA
18 VARDAR SKOPJE
18 DINAMO KIEV
18 RB SALZBURG
18 PARTIZAN

```
18 STURM GRAZ
19 QARABAG FK
19 MALMOE FF
19 FK SUDUVA
19 FC BASEL
19 MOL VIDI FC
19 STURM GRAZ
19 FC ASTANA
19 SLAVIA PRAHA
19 STANDARD LIEGE
19 RED STAR BELGRADE
19 RB SALZBURG
19 PAOK SALONIKI
19 CELTIC FC
19 DINAMO KIEV
19 BATE BORISOV
19 SPARTAK MOSKVA
19 AEK ATHEN
19 FENERBAHCE
19 KF SHKENDIJA 79
19 SPARTAK TRNAVA
19 AFC AJAX
19 PSV EINDHOVEN
19 GALATASARAY
19 VIKTORIA PLZEN
19 CSKA MOSKVA
19 AS ROMA
19 LOKOMOTIV MOSKVA
19 SSC NAPOLI
19 KS PERPARIMI KUKES
19 CFR CLUJ
19 PFC LUDOGORETS RAZGRAD
```

[ ]:
```
data_rosters
```

[ ]:
```
    Year                       Team  ST1  ST2  CM1  CM2  LM  RM  CB1  CB2  LB  \
0     17   SPORTING CLUB DE BASTIA   73   70   74   73  70  75   74   74  73
0     17                 AS MONACO   82   80   83   82  80  77   84   78  77
0     17  FC GIRONDINS DE BORDEAUX   77   75   78   80  77  75   76   74  73
0     17           MONTPELLIER HSC   76   74   80   73  75  77   79   75  75
0     17                 DIJON FCO   74   71   74   73  73  72   71   70  71
..   …                         …    …    …    …    …  ..  ..    …    …  ..
0     19                     INTER   87   82   81   85  81  85   86   84  76
0     19            CLUB BRUGGE KV   78   76   79   78  74  75   76   76  74
0     19          SHAKHTAR DONETSK   76   74   82   80  82  74   75   71  70
0     19                  JUVENTUS   94   85   89   86  85  84   90   86  86
0     19                  FC PORTO   81   85   78   83  78  82   85   83  84
```

```
    RB  GK                      Team_year
0   73  78   SPORTING CLUB DE BASTIA_17
0   79  84                AS MONACO_17
0   76  81  FC GIRONDINS DE BORDEAUX_17
0   73  75         MONTPELLIER HSC_17
0   72  76                DIJON FCO_17
..  ..  ..                          …
0   81  88                    INTER_19
0   74  71          CLUB BRUGGE KV_19
0   75  77         SHAKHTAR DONETSK_19
0   83  86                 JUVENTUS_19
0   80  83                 FC PORTO_19

[268 rows x 14 columns]
```

[ ]:
```
data_full = pd.merge(left=data_games, right=data_rosters, how='inner',␣
↪left_on="home_team_year", right_on="Team_year")
data_full = pd.merge(left=data_full, right=data_rosters, how='inner',␣
↪left_on="away_team_year", right_on="Team_year")
data_full.drop(["Year_x", "tournament", "Year_y", "home_team", "away_team",␣
↪"home_team_year", "away_team_year", "Team_year_x", "Team_year_y", "Team_x",␣
↪"Team_y", "Year"], axis=1, inplace=True)
data_full = data_full.dropna()

data_full.drop(["home_score", "away_score"], axis=1, inplace=True)
data_full
```

[ ]:
```
      Result  home_1_games_ago  home_2_games_ago  home_3_games_ago  \
0     w_away              1.00              1.00              0.00
1     w_home              1.00              0.00              2.00
2     w_away              4.00              0.00              2.00
3     w_home             -1.00              0.00              0.00
4     w_away              1.00              0.00              0.00
…      …                 …                 …                 …
4532  w_away             -4.00             -4.00             -2.00
4533    draw              3.00              4.00             -1.00
4534  w_away              1.00             -1.00             -2.00
4535  w_home              2.00              2.00              0.00
4539    draw             -2.00             -1.00              2.00

      home_4_games_ago  home_5_games_ago  away_1_games_ago  away_2_games_ago  \
0                -1.00             -1.00              4.00              0.00
1                -5.00              1.00              3.00              1.00
2                 1.00              0.00              1.00              2.00
3                 2.00              0.00              2.00              1.00
4                 2.00             -1.00              0.00              2.00
```

```
…               …               …               …               …
4532          1.00           -2.00            0.00            6.00
4533          1.00           -1.00            0.00            0.00
4534         -4.00           -3.00            3.00            1.00
4535          2.00            0.00            0.00           -1.00
4539         -2.00           -5.00            1.00            2.00

      away_3_games_ago  away_4_games_ago  away_5_games_ago  ST1_x  ST2_x  \
0                 4.00              4.00              6.00     73     70
1                 4.00              0.00              4.00     82     80
2                 0.00              2.00              1.00     77     75
3                 2.00              4.00              1.00     76     74
4                 1.00              5.00             -1.00     74     71
…               …               …               …     …     …
4532              1.00              3.00             -2.00     78     73
4533              2.00             -3.00              0.00     84     81
4534              3.00              1.00              3.00     79     78
4535              1.00             -2.00              3.00     90     83
4539              0.00              5.00             -2.00     73     74

      CM1_x  CM2_x  …  CB2_x  LB_x  RB_x  GK_x  ST1_y  ST2_y  CM1_y  CM2_y  \
0        74     73  …     74    73    73    78     86     72     86     85
1        83     82  …     78    77    79    84     86     72     86     85
2        78     80  …     74    73    76    81     86     72     86     85
3        80     73  …     75    75    73    75     86     72     86     85
4        74     73  …     70    71    72    76     86     72     86     85
…     …    …   …    …   …   …   …   …   …   …   …   …
4532     79     79  …     76    75    75    69     82     82     76     79
4533     83     79  …     77    80    78    83     82     82     76     79
4534     79     79  …     80    75    75    82     82     82     76     79
4535     87     84  …     84    84    86    88     78     77     84     80
4539     71     68  …     66    73    77    73     72     73     78     75

      LM_y  RM_y  CB1_y  CB2_y  LB_y  RB_y  GK_y
0       86    83     89     83    82    82    82
1       86    83     89     83    82    82    82
2       86    83     89     83    82    82    82
3       86    83     89     83    82    82    82
4       86    83     89     83    82    82    82
…    …    …     …    …   …    …   …
4532    80    76     79     78    76    79    82
4533    80    76     79     78    76    79    82
4534    80    76     79     78    76    79    82
4535    76    82     82     80    82    79    79
4539    73    73     75     74    70    70    76

[4398 rows x 33 columns]
```

[ ]:
```
data_full.to_csv("data_full_ligue_1.csv", index=False)
```

5 6. Predictions

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

[ ]:
```
X_train, X_test, Y_train, Y_test = train_test_split(data_full.drop("Result",␣
↪axis=1), data_full["Result"], test_size=0.2)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[ ]:
```
# We can change the multi_class
model = LogisticRegression(multi_class="multinomial", max_iter=5000)
```

[ ]:
```
model.fit(X_train, Y_train)
```

[ ]:
```
LogisticRegression(max_iter=5000, multi_class='multinomial')
```

[ ]:
```
model.score(X_train, Y_train), model.score(X_test, Y_test)
```

[ ]:
```
(0.5463331438317226, 0.525)
```

[ ]:
```
from lazypredict.Supervised import LazyClassifier
```

[ ]:
```
clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None)
models, predictions = clf.fit(X_train, X_test, Y_train, Y_test)
models.sort_values("Accuracy", ascending=False)
```

```
100%|          | 29/29 [00:44<00:00,  1.52s/it]
```

[ ]:
```
                               Accuracy  Balanced Accuracy ROC AUC  F1 Score  \
Model
CalibratedClassifierCV             0.53               0.45    None      0.45
SVC                                0.53               0.44    None      0.45
LogisticRegression                 0.53               0.45    None      0.46
LinearDiscriminantAnalysis         0.52               0.45    None      0.46
RidgeClassifier                    0.52               0.44    None      0.45
RidgeClassifierCV                  0.52               0.44    None      0.45
LinearSVC                          0.52               0.44    None      0.45
AdaBoostClassifier                 0.51               0.44    None      0.46
LGBMClassifier                     0.51               0.45    None      0.48
RandomForestClassifier             0.51               0.44    None      0.46
BernoulliNB                        0.50               0.48    None      0.50
```

```
ExtraTreesClassifier              0.50                0.43     None      0.45
QuadraticDiscriminantAnalysis     0.49                0.44     None      0.48
SGDClassifier                     0.49                0.43     None      0.46
GaussianNB                        0.49                0.48     None      0.50
NearestCentroid                   0.49                0.48     None      0.49
NuSVC                             0.48                0.43     None      0.46
KNeighborsClassifier              0.47                0.46     None      0.48
BaggingClassifier                 0.46                0.42     None      0.45
DummyClassifier                   0.45                0.33     None      0.28
LabelSpreading                    0.44                0.41     None      0.44
LabelPropagation                  0.44                0.41     None      0.44
ExtraTreeClassifier               0.42                0.39     None      0.42
Perceptron                        0.39                0.36     None      0.39
DecisionTreeClassifier            0.38                0.35     None      0.38
PassiveAggressiveClassifier       0.36                0.36     None      0.37

                                Time Taken
Model
CalibratedClassifierCV               17.79
SVC                                   2.94
LogisticRegression                    0.20
LinearDiscriminantAnalysis            0.13
RidgeClassifier                       0.08
RidgeClassifierCV                     0.09
LinearSVC                             3.98
AdaBoostClassifier                    1.55
LGBMClassifier                        1.16
RandomForestClassifier                2.18
BernoulliNB                           0.09
ExtraTreesClassifier                  2.16
QuadraticDiscriminantAnalysis         0.08
SGDClassifier                         0.47
GaussianNB                            0.08
NearestCentroid                       0.08
NuSVC                                 4.06
KNeighborsClassifier                  0.27
BaggingClassifier                     1.09
DummyClassifier                       0.06
LabelSpreading                        2.81
LabelPropagation                      2.04
ExtraTreeClassifier                   0.10
Perceptron                            0.10
DecisionTreeClassifier                0.25
PassiveAggressiveClassifier           0.10
```

[ ]: