Predikcija ishoda NBA utakmica

NBA

- Profesionalna američka košarkaška liga
- 30 timova podeljenih u dve konferencije (istočna i zapadna)
- Jedna sezona se sastoji od dva dela: Regularni deo i Doigravanje (Play-off)
- Tokom regularnog dela sezone odigra se 1230 utakmica (svaki tim po 82)
- Napredna statistika

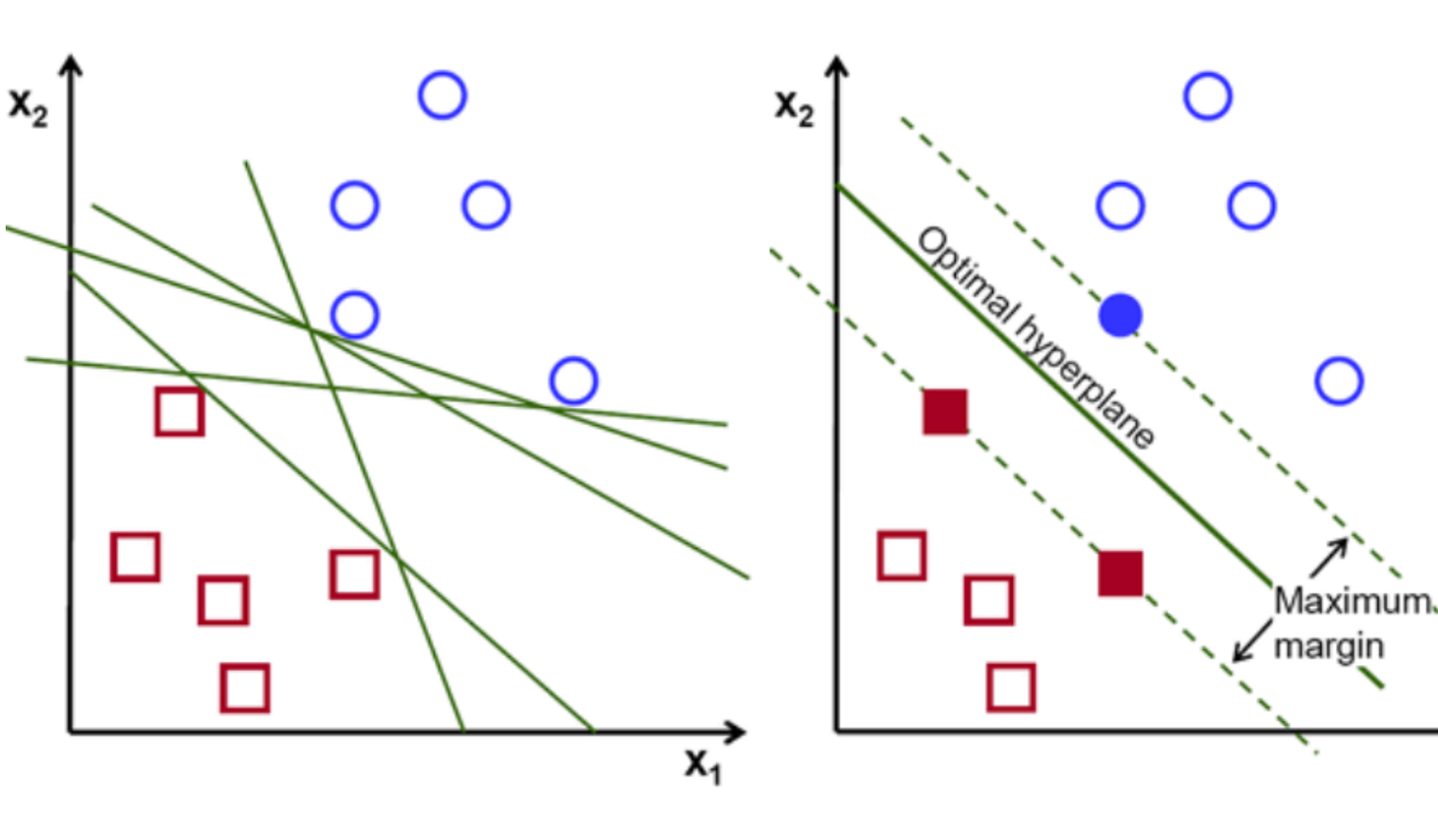
Zadatak: Predvideti da li u NBA utakmici pobeđuje domaćin ili gost

- Koriste se podaci dostupni pre odigravanja utakmice koja se predviđa kao i statistika predviđane utakmice
- Za trening kao i za predviđanje koriste se isključivo utakmice regularnog dela sezone
- Koršćeni su kasifikacioni algoritmi SVM (Support Vector Machine), Random Forest, Naive Bayes, kao i regresioni algoritam Linear Regression

Algoritmi

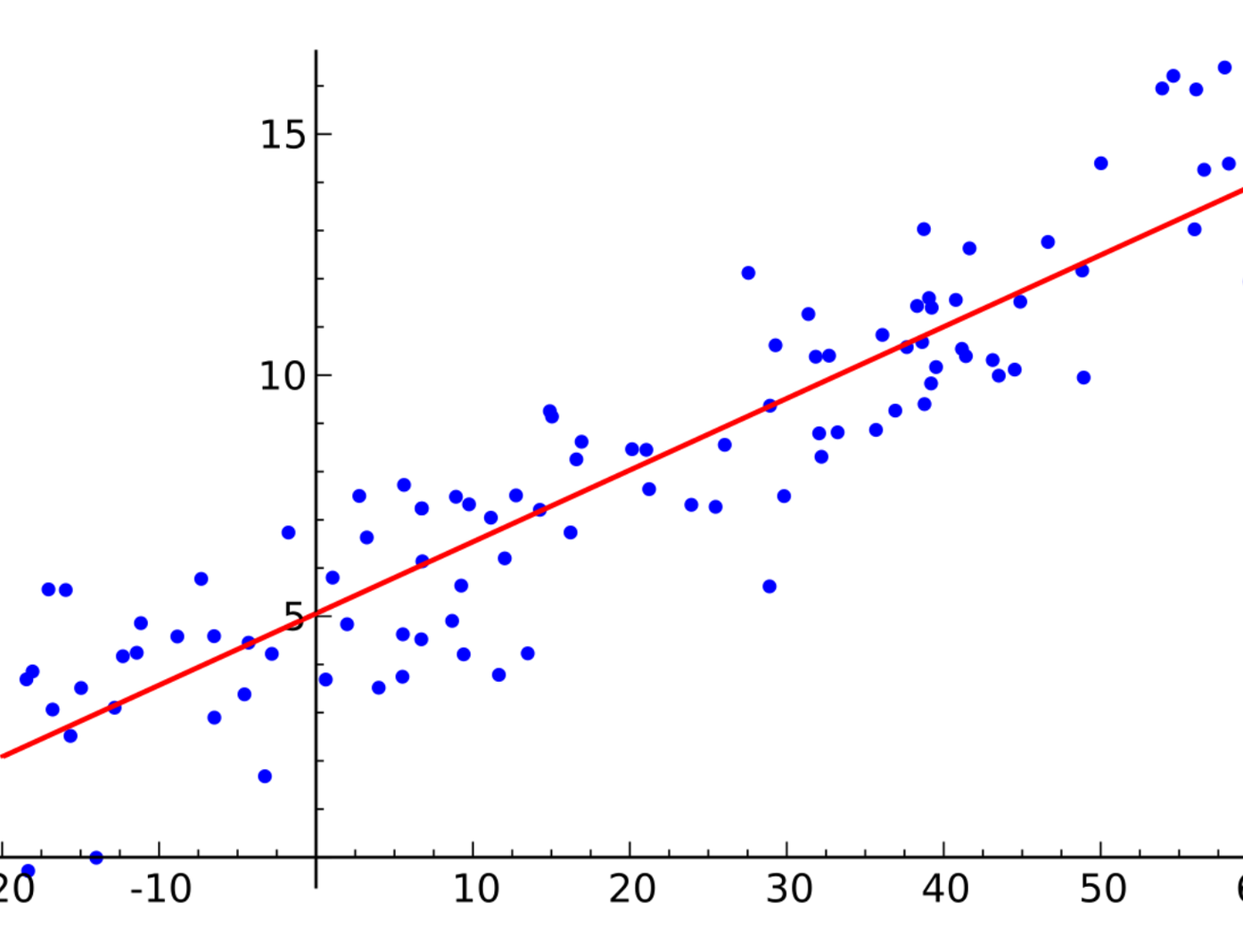
Suport Vector Machine (SVM)

- Klasifikacioni algoritam nadgledanog učenja koji za cilj ima pronalaženje optimalne granice (hyperplane) u N-dimenzionalnom prostoru, koja klasifikuje tačke podataka.
- Dimenzionalnost prostora zavisi broja feature-a (npr. ako je broj feature-a 2, hyperplane je obična linija).



Linear Regression

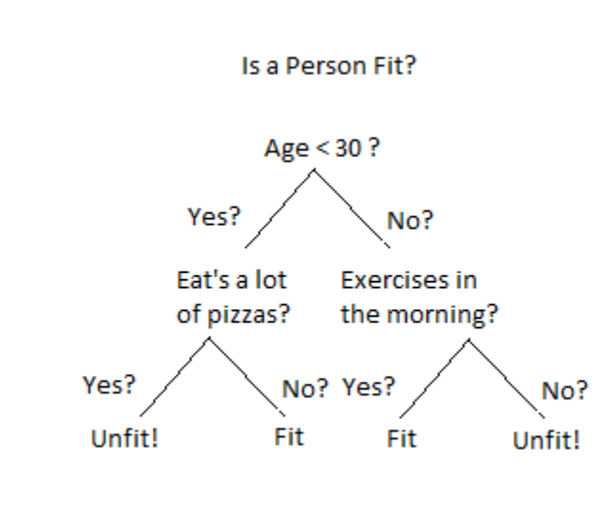
- Algoritam nadgledanog učenja čiji je zadatak da predvidi vrednost zavisne promenljive (y) na osnovu vrednosti nezavisnih promenljivih (x).
- Traži se veza između y i x.
- Zavisna promenljiva (y) mora biti kontinualna.



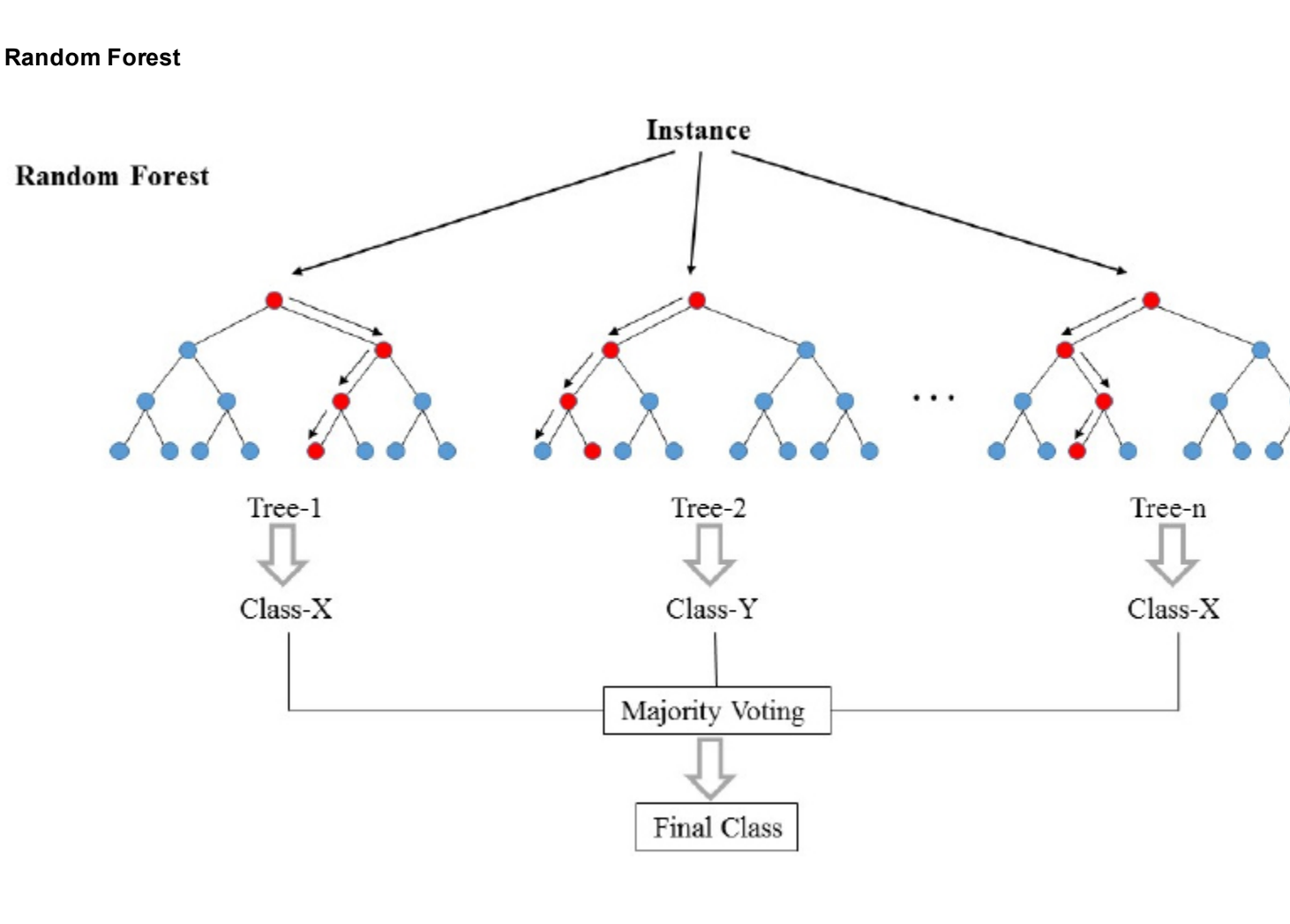
Random Forest

- Klasifikacioni algoritam koji se bazira na upotrebi **stabala odluke** (decision trees), koja zajedno prediktuju klasu.
- Svako stablo izbacuje svoju procenu klase, pa se klasa sa najviše glasova bira kao prediktovana.

Decision tree



Random Forest



Naive Bayes

- Probabilistički algoritam mašinskog učenja koji služi za klasifikaciju.
- Za osnovu koristi Bajesovu teoremu.
- Bajesova teorema pronalazi verovatnoću da će se događaj A desiti, ako se događaj B već dogodio.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Podaci

Podaci za ovaj projekat su prikupljeni sa sledećeg sajta:

- https://www.kaggle.com

Originalni dataset pronađen je na sajtu Kaggle (može se preuzeti ovde), u CSV formatu. Pokriva sve odigrane utakmice u periodu od 2012. do 2018. godine (6 sezona).

Za ovaj projekat, sezone su iskorišćene na sledeći način:

- Trening sezone: 2014/15, 2015/16, 2016/17
- Test sezona: 2017/18

Iskorišćeni dataset sadrži osnovne i napredne statistike za svaku odigranu utakmicu. Značenja i objašnjenja svakog statističkog parametra (feature-a) mogu se pogedati na sledećim linkovima:

- https://basketball.realgm.com/info/glossary
- https://www.nbastuffer.com/team-evaluation-metrics/

Organizacija projekta

Projekat se sastoji iz tri python (.py) fajla:

- **main.py** - učitavanje dataset-a i pozivanje kreiranih metoda
- **dataPreparation.py** - klasa DataPreparation za obradu dataset-a i dodavanje novih feature-a
- **MLAlgorithms.py** - klasa MLAlgorithms koja sadrži metode algoritama za predikciju.

Implementacija

main.py

In [1]:
```python
import pandas as pd
```

Učitavanje data set-a za trening (sezone 2014-2017):

In [2]:
```python
data_frame = pd.read_csv("C:\\Users\\Zola\\Desktop\\NBA14-17.csv", parse_dates=["gmDate"])
data_frame.head(3)
```

Out[2]:

	gmDate	team	teamConf	teamDiv	teamLoc	teamRslt	teamMin	teamDayOff	teamPTS	teamAST	...	opptFIC40	opptO
0	2014-10-28	ORL	East	Southeast	Away	Loss	240	0	84	17	...	80.1042	106.0
1	2014-10-28	NO	West	Southwest	Home	Win	240	0	101	20	...	49.6875	88.2
2	2014-10-28	DAL	West	Southwest	Away	Loss	240	0	100	17	...	61.0417	116.6

3 rows × 115 columns

In [3]:
```python
data_frame.tail(3)
```

Out[3]:

	gmDate	team	teamConf	teamDiv	teamLoc	teamRslt	teamMin	teamDayOff	teamPTS	teamAST	...	opptFIC40	op
7377	2017-12-04	LAC	West	Pacific	Home	Win	241	2	115	29	...	57.0833	10
7378	2017-12-04	NO	West	Southwest	Away	Win	241	1	103	20	...	57.1875	10
7379	2017-12-04	POR	West	Northwest	Home	Loss	240	2	100	22	...	65.0415	10

3 rows × 115 columns

Učitavanje data set-a za trening (sezona 2017/18):

In [4]:
```python
test_data_frame = pd.read_csv("C:\\Users\\Zola\\Desktop\\NBA17-18.csv", parse_dates=["gmDate"])
test_data_frame.tail(3)
```

Out[4]:

	gmDate	team	teamConf	teamDiv	teamLoc	teamRslt	teamMin	teamDayOff	teamPTS	teamAST	...	opptFIC40	op
2457	2018-11-04	POR	West	Northwest	Home	Win	240	2	102	19	...	54.7718	9
2458	2018-11-04	HOU	West	Southwest	Away	Loss	241	1	83	11	...	65.8333	10
2459	2018-11-04	SAC	West	Pacific	Home	Win	240	2	96	22	...	34.8548	9

3 rows × 115 columns

dataPreparation.py

In [5]:
```python
from collections import defaultdict
```

Metoda za obradu data set-a:

In [9]:
```python
    def prepare_data_frame(df, train):
        # Uklanjanje duplikata utakmica
        index_to_drop = df[df["teamLoc"] == "Away"].index
        df.drop(index_to_drop, inplace=True)
        df.reset_index(inplace=True)
        df.drop(["index"], axis=1, inplace=True)

        # Promena naziva kolona
        df.rename(columns={'team': 'homeTeam', 'opponent': 'awayTeam'}, inplace=True)

        # Dodavanje "homeWin" kolone --> label
        df["homeWin"] = df["teamRslt"] == "Win"

        # Procenat pobeda domacina
        n_games = df["homeWin"].count()
        n_homewins = df["homeWin"].sum()
        win_percentage = n_homewins / n_games
        if train:
            print("Home Win percentage in train seasons (from data): {0: .2f}%".format(100 * win_per
centage))
        else:
            print("Home Win percentage in test seasons (from data): {0: .2f}%".format(100 * win_perc
entage))

        # Dodavanje "ptsDifference" kolone --> label
        df["ptsDifference"] = df["teamPTS"] - df["opptPTS"]

        last_match_result(df)
        streak(df)
        last_between_two(df)
```

Sređivanje trening data set-a:

In [10]:
```python
prepare_data_frame(data_frame, True)
print(data_frame.columns)
```

```
Home Win percentage in train seasons (from data):  58.24%
Index(['gmDate', 'homeTeam', 'teamConf', 'teamDiv', 'teamLoc', 'teamRslt',
       'teamMin', 'teamDayOff', 'teamPTS', 'teamAST',
       ...
       'opptSTL/TO', 'poss', 'pace', 'homeWin', 'ptsDifference',
       'homeTeamLastWin', 'awayTeamLastWin', 'homeTeamWinStreak',
       'awayTeamWinStreak', 'homeTeamWonLast'],
      dtype='object', length=122)
```

Sređivanje test data set-a:

In [11]:
```python
prepare_data_frame(test_data_frame, False)
print(test_data_frame.columns)
```

```
Home Win percentage in test seasons (from data):  57.89%
Index(['gmDate', 'homeTeam', 'teamConf', 'teamDiv', 'teamLoc', 'teamRslt',
       'teamMin', 'teamDayOff', 'teamPTS', 'teamAST',
       ...
       'opptSTL/TO', 'poss', 'pace', 'homeWin', 'ptsDifference',
       'homeTeamLastWin', 'awayTeamLastWin', 'homeTeamWinStreak',
       'awayTeamWinStreak', 'homeTeamWonLast'],
      dtype='object', length=122)
```

Kreiranje novih feature-a

Da li je tim pobedio ili izgubio poslednje odigranu utakmicu?

In [6]:
```python
    def last_match_result(df):

        # Rečnik za čuvanje rezultata utakmice koju je tim poslednju odigrao
        won_last = {
            "ATL": False, "BOS": False, "BKN": False, "CHA": False, "CHI": False, "CLE": False,
            "DAL": False, "DEN": False, "DET": False, "GS": False, "HOU": False, "IND": False,
            "LAC": False, "LAL": False, "MEM": False, "MIA": False, "MIL": False, "MIN": False,
            "NO": False, "NY": False, "OKC": False, "ORL": False, "PHI": False, "PHO": False,
            "POR": False, "SAC": False, "SA": False, "TOR": False, "UTA": False, "WAS": False
        }

        df["homeTeamLastWin"] = False
        df["awayTeamLastWin"] = False

        for index_team, row_df in df.iterrows():
            team_h = row_df["homeTeam"]
            team_a = row_df["awayTeam"]
            row_df["homeTeamLastWin"] = won_last[team_h]
            row_df["awayTeamLastWin"] = won_last[team_a]
            df.iloc[index_team] = row_df
            if row_df["teamPTS"] > row_df["opptPTS"]:
                won_last[team_h] = True
            else:
                won_last[team_h] = False

            won_last[team_a] = not won_last[team_h]
```

In [12]:
```python
print(data_frame["homeTeamLastWin"].tail())
```

```
3685    False
3686     True
3687    False
3688     True
3689     True
Name: homeTeamLastWin, dtype: bool
```

In [13]:
```python
print(data_frame["awayTeamLastWin"].tail())
```

```
3685    False
3686    False
3687     True
3688     True
3689    False
Name: awayTeamLastWin, dtype: bool
```

Koliko je utakmica u nizu tim pobedio?

In [7]:
```python
    def streak(df):

        df["homeTeamWinStreak"] = 0
        df["awayTeamWinStreak"] = 0

        win_streak = {
            "ATL": 0, "BOS": 0, "BKN": 0, "CHA": 0, "CHI": 0, "CLE": 0,
            "DAL": 0, "DEN": 0, "DET": 0, "GS": 0, "HOU": 0, "IND": 0,
            "LAC": 0, "LAL": 0, "MEM": 0, "MIA": 0, "MIL": 0, "MIN": 0,
            "NO": 0, "NY": 0, "OKC": 0, "ORL": 0, "PHI": 0, "PHO": 0,
            "POR": 0, "SAC": 0, "SA": 0, "TOR": 0, "UTA": 0, "WAS": 0
        }

        for index, row in df.iterrows():
            team = row["homeTeam"]
            opponent = row["awayTeam"]
            row["homeTeamWinStreak"] = win_streak[team]
            row["awayTeamWinStreak"] = win_streak[opponent]
            df.iloc[index] = row
            if row["teamPTS"] > row["opptPTS"]:
                win_streak[team] += 1
                win_streak[opponent] = 0
            else:
                win_streak[opponent] += 1
                win_streak[team] = 0
```

In [14]:
```python
print(data_frame["homeTeamWinStreak"].tail())
```

```
3685    0
3686    1
3687    0
3688    6
3689    3
Name: homeTeamWinStreak, dtype: int64
```

In [15]:
```python
print(data_frame["awayTeamWinStreak"].tail())
```

```
3685    0
3686    0
3687    5
3688    1
3689    0
Name: awayTeamWinStreak, dtype: int64
```

Ko je pobedio u poslednje odigranom duelu dva konkretna tima?

In [8]:
```python
    def last_between_two(df):

        last_match_winner = defaultdict(int)

        def home_team_won_last(row):
            team = row["homeTeam"]
            opponent = row["awayTeam"]
            teams = tuple(sorted([team, opponent]))
            result = 1 if last_match_winner[teams] == row["homeTeam"] else 0
            winner = row["homeTeam"] if row["homeWin"] else row["awayTeam"]
            last_match_winner[teams] = winner
            return result

        df["homeTeamWonLast"] = df.apply(home_team_won_last, axis=1)
```

In [16]:
```python
print(data_frame["homeTeamWonLast"].tail())
```

```
3685    1
3686    0
3687    1
3688    0
3689    0
Name: homeTeamWonLast, dtype: int64
```

MLAlgorithms.py

In [17]:
```python
import matplotlib.pyplot as plt
import seaborn as seabornInstance
import shap
from sklearn import metrics
from sklearn import svm
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import MinMaxScaler
```

Izbacivanje feature-a nepotrebnih za predikciju:

In [18]:
```python
    def drop_features(df):
        fields_to_drop = ["gmDate", "homeTeam", "teamConf", "teamDiv", "teamLoc", "teamRslt",
                          "awayTeam", "opptConf", "opptDiv", "opptLoc", "opptRslt", "ptsDifference",
"homeWin"]
        X = df.drop(fields_to_drop, axis=1).dropna()
        return X
```

SVM

In [21]:
```python
    def svm_func(df, test_df):

        X_train = drop_features(df)
        y_train = df["homeWin"].values

        scaler = MinMaxScaler(feature_range=(-1, 1))
        X_train = scaler.fit_transform(X_train)

        X_test = drop_features(test_df)
        y_test = test_df["homeWin"].values
        X_test = scaler.fit_transform(X_test)

        clf = svm.SVC(kernel="linear")

        clf.fit(X_train, y_train)

        pred = clf.predict(X_test)

        print("metrics.accuracy SVM: " + str(metrics.accuracy_score(y_test, pred)))
        print("Confusion matrix for SVM: \n")
        print(confusion_matrix(y_test, pred))
        print("Classification report for SVM: \n")
        print(classification_report(y_test, pred))
        print("\n\n")
```

In [22]:
```python
svm_func(data_frame, test_data_frame)
```

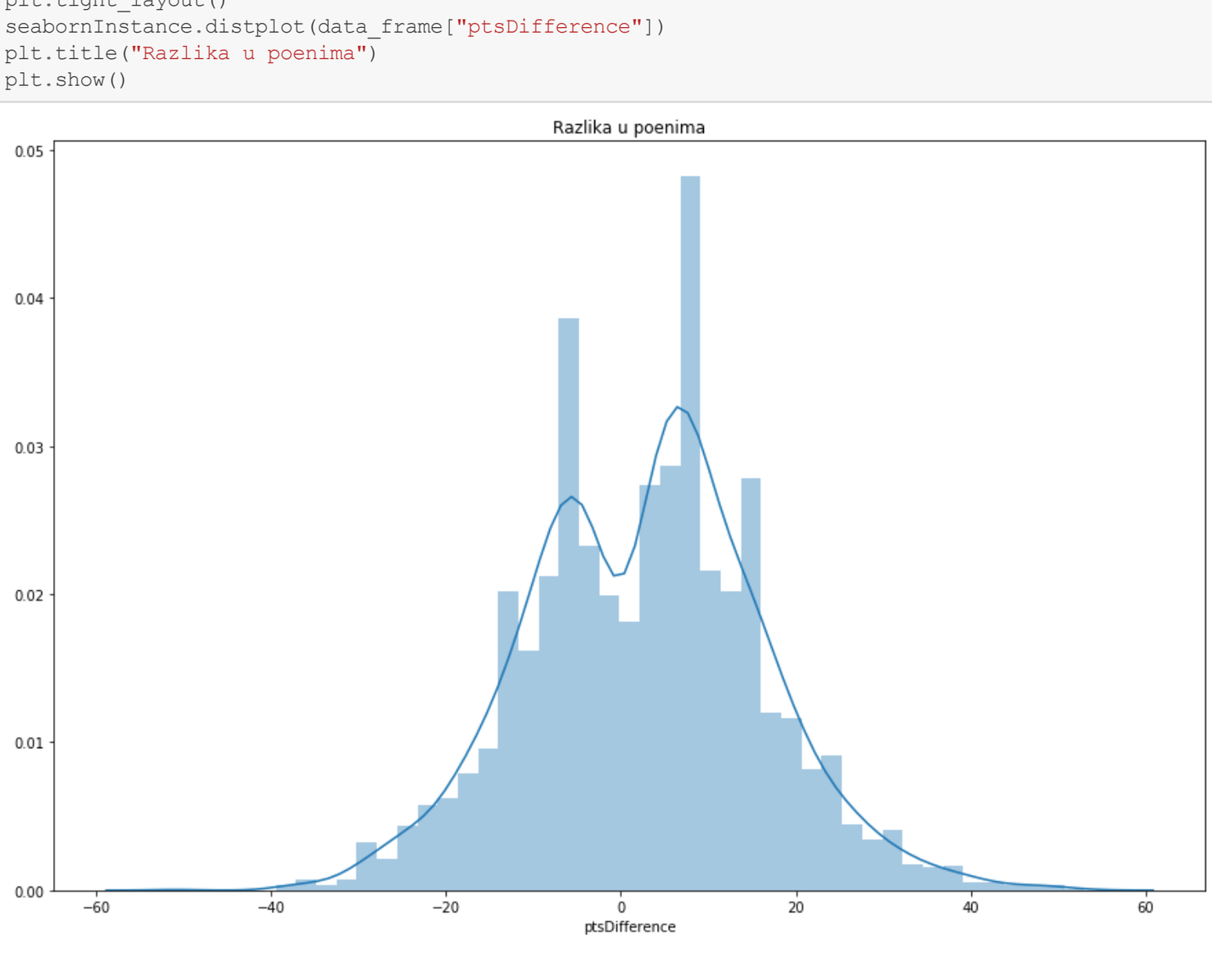
```
metrics.accuracy SVM: 0.9926829268292683
Confusion matrix for SVM:

[[517   1]
 [  8 704]]
Classification report for SVM:

              precision    recall  f1-score   support

       False       0.98      1.00      0.99       518
        True       1.00      0.99      0.99       712

    accuracy                           0.99      1230
   macro avg       0.99      0.99      0.99      1230
weighted avg       0.99      0.99      0.99      1230
```

Linear Regression

In [23]:
```python
    X_train = drop_features(data_frame)
    y_train = data_frame["ptsDifference"].values

    X_test = drop_features(test_data_frame)
    y_test = test_data_frame["ptsDifference"]
```

Kojom razlikom se najčešće završavaju utakmice?

In [24]:
```python
# To see max average points difference
plt.figure(figsize=(15, 10))
plt.tight_layout()
seabornInstance.distplot(data_frame["ptsDifference"])
plt.title("Razlika u poenima")
plt.show()
```



Vidimo da je najčešća razlika na kraju meča oko +8 (+ znači da je pobedio domaćin, - da je pobedio gost).

Kreiranje i treniranje regresora:

In [25]:
```python
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[25]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Optimalni koeficijenti za atribute:

In [26]:
```python
print("Linear Regression parameters: \n")
coeff_df = pd.DataFrame(regressor.coef_, X_test.columns, columns=['Coefficient'])
print(coeff_df.head())
```

```
Linear Regression parameters:

             Coefficient
teamMin    -6.408958e-15
teamDayOff  1.942890e-16
teamPTS     8.419867e-01
teamAST    -1.580133e-01
teamTO      1.580133e-01
```

Predikcija i prikaz stvarnih i prediktovanih vrednosti razlike u poenima:

In [27]:
```python
y_pred = regressor.predict(X_test)
df1 = pd.DataFrame({"Stvarni rezultat": y_test, "Prediktovani rezultat": y_pred})
print("Prikaz stvarne i prediktovane razlike u poenima: \n")
print(df1.head())
```

```
Prikaz stvarne i prediktovane razlike u poenima:

   Stvarni rezultat  Prediktovani rezultat
0                 3                    3.0
1                -1                   -1.0
2                12                   12.0
3                 9                    9.0
4                 7                    7.0
```

R2 skor za algoritam Linear Regression:

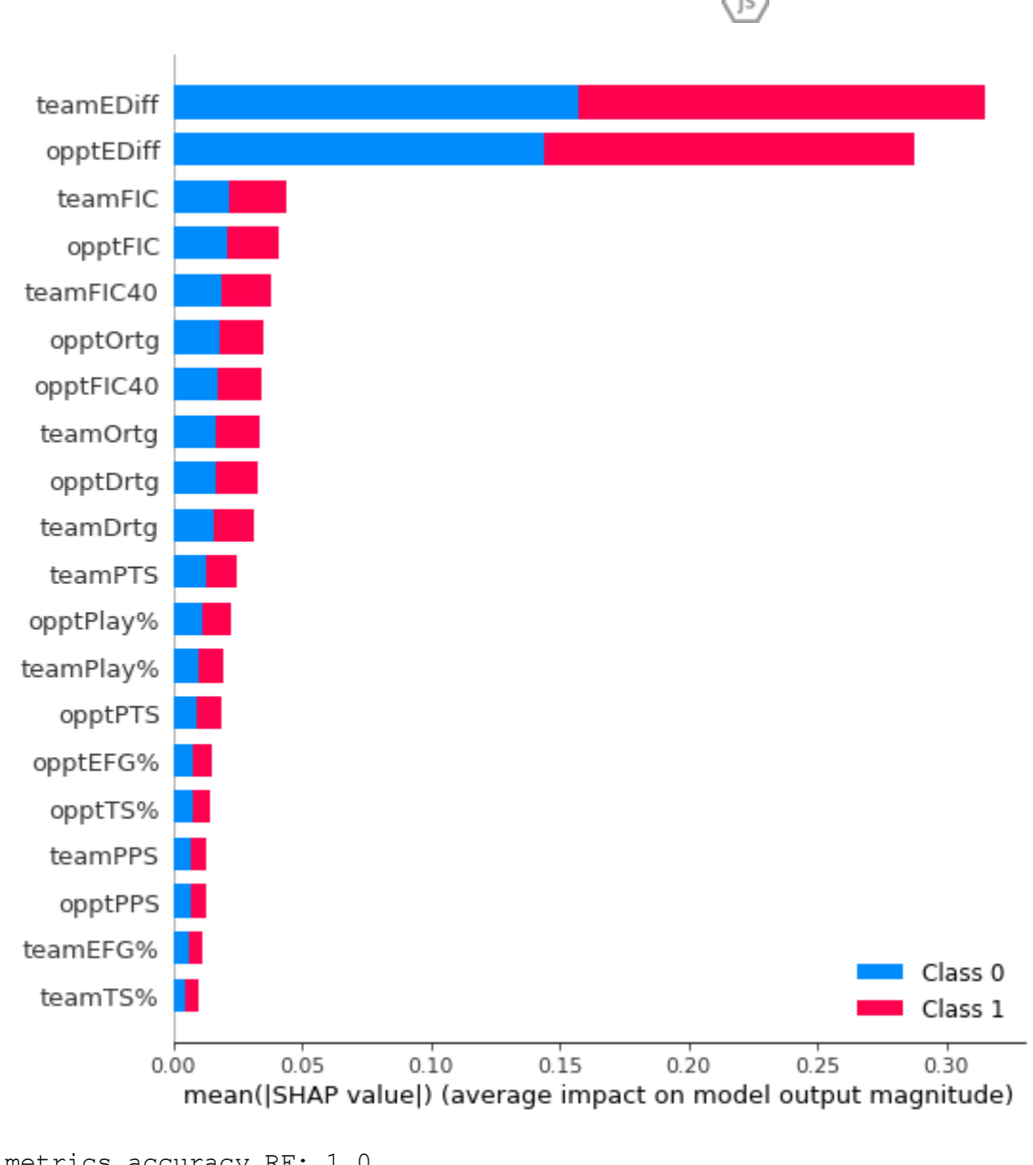
In [28]:
```python
scores2 = regressor.score(X_test, y_test)
print("R2 score LR: " + str(scores2))
```

```
R2 score LR: 0.9999997506522216
```

Random Forest

In [29]:
```python
    def random_forest_func(df, test_df):

        X_train = drop_features(df)
        y_train = df["homeWin"].values

        X_test = drop_features(test_df)
        y_test = test_df["homeWin"]

        rf = RandomForestClassifier(random_state=0, n_estimators=100, bootstrap=True, max_features=
"sqrt")
        rf.fit(X_train, y_train)

        shap.initjs()
        explainer = shap.TreeExplainer(rf)
        shap_values = explainer.shap_values(X_test)
        shap.summary_plot(shap_values, X_test)

        y_pred_rf = rf.predict(X_test)
        print("metrics.accuracy RF: " + str(metrics.accuracy_score(y_test, y_pred_rf)))
        print("Confusion matrix for RF: \n")
        print(confusion_matrix(y_test, y_pred_rf))
        print("Classification report for RF: \n")
        print(classification_report(y_test, y_pred_rf))
```

In [30]:
```python
random_forest_func(data_frame, test_data_frame)
```




```
metrics.accuracy RF: 1.0
Confusion matrix for RF:

[[518   0]
 [  0 712]]
Classification report for RF:

              precision    recall  f1-score   support

       False       1.00      1.00      1.00       518
        True       1.00      1.00      1.00       712

    accuracy                           1.00      1230
   macro avg       1.00      1.00      1.00      1230
weighted avg       1.00      1.00      1.00      1230
```

Naive Bayes

In [31]:
```python
    def naive_bayes(df, test_df):

        X_train = drop_features(df)
        y_train = df["homeWin"].values

        X_test = drop_features(test_df)
        y_test = test_df["homeWin"]

        gnb = GaussianNB()
        gnb.fit(X_train, y_train)
        y_pred_nb = gnb.predict(X_test)
        print("metrics.accuracy NB: " + str(metrics.accuracy_score(y_test, y_pred_nb)))
        print("Confusion matrix for RF: \n")
        print(confusion_matrix(y_test, y_pred_nb))
        print("Classification report for RF: \n")
        print(classification_report(y_test, y_pred_nb))
        print("\n\n")
```

In [32]:
```python
naive_bayes(data_frame, test_data_frame)
```

```
metrics.accuracy NB: 0.9560975609756097
Confusion matrix for RF:

[[495  23]
 [ 31 681]]
Classification report for RF:

              precision    recall  f1-score   support

       False       0.94      0.96      0.95       518
        True       0.97      0.96      0.96       712

    accuracy                           0.96      1230
   macro avg       0.95      0.96      0.96      1230
weighted avg       0.96      0.96      0.96      1230
```

Zaključak

Na osnovu dobijenih rezultata, sva četiri algoritma (3 klasifikaciona i 1 regresioni), primenjena na dati dataset su pokazali dobre rezultate u predviđanju ishoda NBA utakmica.

Algoritam koji se najbolje pokazao je Random Forest sa 100% tačnosti, dok se Naive Bayes najlošije pokazao ali sa opet više nego zadovoljavajućih 95.6% tačnosti.