Pojačavanje (eng. boosting)

Osnovna ideja pojačavanja je da se ansambl gradi dodajući model po model, pri čemu se svaki od modela obučava tako da što bolje nadomesti slabosti tekućeg skupa modela, odnosno da ga pojača. Izgradnja ansambla počinje od modela $F_0(x) = 0$, i pretpostavljajući da je dostupan ansambl F_{m-1} gradi ansambl F_m dodavanjem još jednog modela koji se obučava tako da što bolje predvidi grešku predviđanja postojećeg ansambla. Broj modela od kojih se sastoji ansambl je hiperparametar.

XGBoost

XGBoost je po pitanju preciznosti jedan od najboljih metoda mašinskog učenja.
XGBoost je algoritam gradijentnog pojačavanja pomoću stabala odlučivanja. Osnovna ideja dolazi iz gradijentnih optimizacionih metoda, gde se tekuće rešenje popravlja dodavanjem vektora proporcionalnog negativnoj vrednosti gradijenta funkcije koja se minimizuje.

Za detalje možete pogledati skriptu profesora Mladena Nikolića.

AdaBoost

Ideja AdaBoost algoritma je bazirana na promenljivim težinama instanci. Na početku je potrebno nad skupom za treniranje naučiti model klasifikacije ili regresije (takozvani bazni model) dajući podjednak značaj tj. težine svim instancama. Na primer, ako u skupu za treniranje postoji M instanci mogu im se pridružiti težine $\frac{1}{M}$. Potom je potrebno analizirati greške koje je bazni model napravio i proporcionalno njima povećati težine instancama. Na ovaj način se model usmerava da obrati više pažnje na instance sa većim greškama ne bi li postao precizniji. Nad skupom instanci sa novim težinama trenira se sledeći model, a ceo postupak se ponavlja dok se ne dosegne zadati broj modela ansambla. Ovakav pristup sve vreme podržava pojačavanje jer su narednim modelima poznate greške prethodnih modela.

Za detalje možete pogledati skriptu profesora Mladena Nikolića.

Primer regresije

U nastavku ćemo upoznati XGBoost i AdaBoost modele koje ćemo primeniti u zadatku određivanja cena nekretnina koristeći California Housing skup podataka.

Da bismo mogli da kreiramo XGBoost model, potrebno je da instaliramo XGBoost biblioteku. AdaBoost model se nalazi u paketu ensemble biblioteke sklearn

In [1]:
```
# !pip install xgboost
import xgboost
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn import model_selection
from sklearn import datasets
```

```
from sklearn import preprocessing
from sklearn import metrics
```

In [3]:
```
from sklearn import tree
from sklearn import ensemble
```

Prvo ćemo učitati podatke i pripremiti skup za treniranje i skup za testiranje.

In [4]:
```
data = datasets.fetch_california_housing()
X = data.data
y = data.target
```

In [5]:
```
X_train, X_test, y_train, y_test = model_selection.train_test_split(
    X, y, test_size=0.2, random_state=7)
```

In [6]:
```
scaler = preprocessing.StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Pravimo oba modela

In [7]:
```
model_xgboost = xgboost.XGBRegressor(objective='reg:squarederror',
                                     n_estimators=100, max_depth=5)
model_adaboost = ensemble.AdaBoostRegressor(
    base_estimator=tree.DecisionTreeRegressor(max_depth=5), n_estimators=10, random_stat
# Hiperparametri su broj stabala, maksimalna dubina...
```

In [8]:
```
model_xgboost.fit(X_train, y_train);
```

In [9]:
```
model_adaboost.fit(X_train, y_train);
```

In [10]:
```
y_predicted1 = model_xgboost.predict(X_test)
```

In [11]:
```
metrics.mean_squared_error(y_test, y_predicted1)
```

Out[11]:
```
0.23260912437293624
```

In [12]:
```
metrics.r2_score(y_test, y_predicted1)
```

Out[12]:
```
0.8270585337255897
```

In [13]:
```
y_predicted2 = model_adaboost.predict(X_test)
```

In [14]:
```
metrics.mean_squared_error(y_test, y_predicted2)
```

Out[14]:
```
0.43324252114716993
```

In [15]:
```
metrics.r2_score(y_test, y_predicted2)
```

Out[15]:
```
0.6778905511054352
```

XGBoost je dao dosta bolje rezultate. AdaBoost je u ovom zadatku imao i problem sa preprilagodjavanjem, jer je npr. model sa 100 stabala davao izuzetno loše rezultate:

In [16]:
```
model_adaboost2 = ensemble.AdaBoostRegressor(
    base_estimator=tree.DecisionTreeRegressor(max_depth=5),
```

```
    n_estimators=100, random_state=7)
model_adaboost2.fit(X_train, y_train);
```

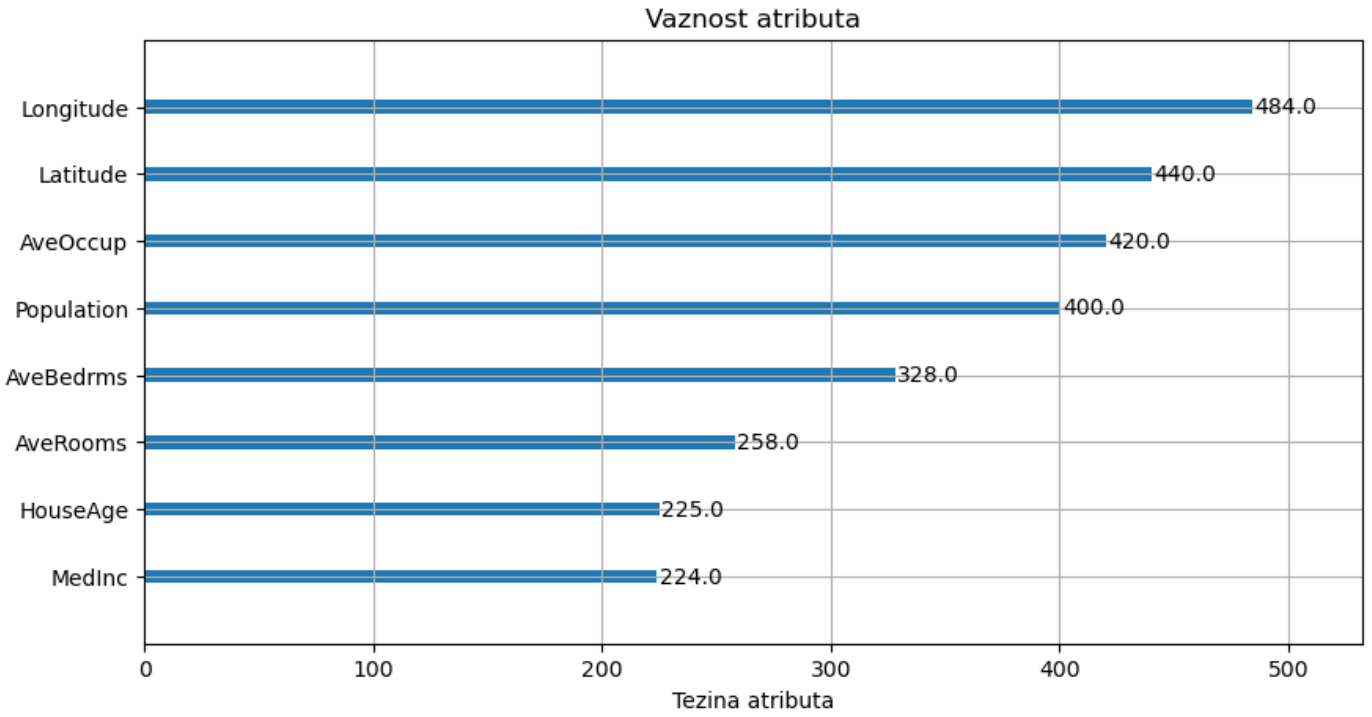
In [17]:
```
metrics.r2_score(y_test, model_adaboost2.predict(X_test))
```

Out[17]:
```
0.3503799020678511
```

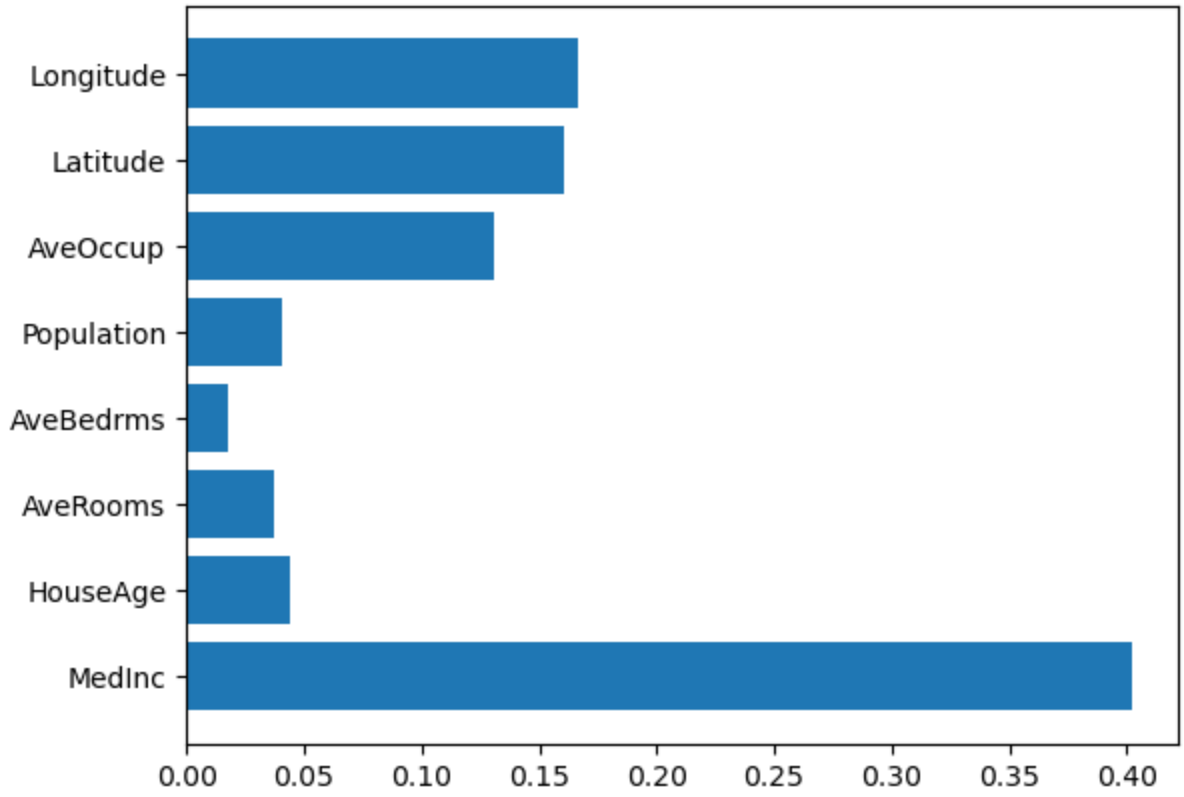
Za oba modela možemo dobiti informaciju o važnosti atributa.

In [18]:
```
plt.figure(figsize=(10, 5))
ax = plt.subplot(1, 1, 1)
xgboost.plot_importance(model_xgboost, ax= ax, xlabel='Tezina atributa', ylabel=None,
title='Vaznost atributa').set_yticklabels(data.feature_names)
```

Out[18]:
```
[Text(0, 0, 'MedInc'),
 Text(0, 1, 'HouseAge'),
 Text(0, 2, 'AveRooms'),
 Text(0, 3, 'AveBedrms'),
 Text(0, 4, 'Population'),
 Text(0, 5, 'AveOccup'),
 Text(0, 6, 'Latitude'),
 Text(0, 7, 'Longitude')]
```



In [19]:
```
plt.barh(data.feature_names, model_adaboost.feature_importances_)
plt.show()
```


Longitude
Latitude
AveOccup
Population
AveBedrms
AveRooms
HouseAge
MedInc
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40

In [ ]: