hitters

June 18, 2024

[35]:
```
# Abdurrahman Bulut
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor,␣
↪AdaBoostRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import GridSearchCV
import joblib
```

[2]:
```
df = pd.read_csv("hitters.csv")
```

[3]:
```
df.head()
```

[3]:

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	\
0	293	66	1	30	29	14	1	293	66	1	30	
1	315	81	7	24	38	39	14	3449	835	69	321	
2	479	130	18	66	72	76	3	1624	457	63	224	
3	496	141	20	65	78	37	11	5628	1575	225	828	
4	321	87	10	39	42	30	2	396	101	12	48	

	CRBI	CWalks	League	Division	PutOuts	Assists	Errors	Salary	NewLeague
0	29	14	A	E	446	33	20	NaN	A
1	414	375	N	W	632	43	10	475.0	N
2	266	263	A	W	880	82	14	480.0	A
3	838	354	N	E	200	11	3	500.0	N
4	46	33	N	E	805	40	4	91.5	N

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 322 entries, 0 to 321
Data columns (total 20 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   AtBat      322 non-null    int64
 1   Hits       322 non-null    int64
 2   HmRun      322 non-null    int64
 3   Runs       322 non-null    int64
 4   RBI        322 non-null    int64
 5   Walks      322 non-null    int64
 6   Years      322 non-null    int64
 7   CAtBat     322 non-null    int64
 8   CHits      322 non-null    int64
 9   CHmRun     322 non-null    int64
 10  CRuns      322 non-null    int64
 11  CRBI       322 non-null    int64
 12  CWalks     322 non-null    int64
 13  League     322 non-null    object
 14  Division   322 non-null    object
 15  PutOuts    322 non-null    int64
 16  Assists    322 non-null    int64
 17  Errors     322 non-null    int64
 18  Salary     263 non-null    float64
 19  NewLeague  322 non-null    object
dtypes: float64(1), int64(16), object(3)
memory usage: 50.4+ KB
```

[5]:
```
# Numerik ve kategorik değişkenleri belirleyelim
numerik_degiskenler = df.select_dtypes(include=['int64', 'float64']).columns.
  ↪tolist()
kategorik_degiskenler = df.select_dtypes(include=['object']).columns.tolist()
```

[6]:
```
print("Numerik Değişkenler:", numerik_degiskenler)
print("Kategorik Değişkenler:", kategorik_degiskenler)
```

```
Numerik Değişkenler: ['AtBat', 'Hits', 'HmRun', 'Runs', 'RBI', 'Walks', 'Years',
'CAtBat', 'CHits', 'CHmRun', 'CRuns', 'CRBI', 'CWalks', 'PutOuts', 'Assists',
'Errors', 'Salary']
Kategorik Değişkenler: ['League', 'Division', 'NewLeague']
```

[7]:
```
df.isnull().sum()
```

[7]:
```
AtBat        0
Hits         0
HmRun        0
Runs         0
RBI          0
```

```
Walks          0
Years          0
CAtBat         0
CHits          0
CHmRun         0
CRuns          0
CRBI           0
CWalks         0
League         0
Division       0
PutOuts        0
Assists        0
Errors         0
Salary        59
NewLeague      0
dtype: int64
```

[8]:
```
# eksik maaşlar silindi
df = df.dropna()
```

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 263 entries, 1 to 321
Data columns (total 20 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   AtBat      263 non-null    int64
 1   Hits       263 non-null    int64
 2   HmRun      263 non-null    int64
 3   Runs       263 non-null    int64
 4   RBI        263 non-null    int64
 5   Walks      263 non-null    int64
 6   Years      263 non-null    int64
 7   CAtBat     263 non-null    int64
 8   CHits      263 non-null    int64
 9   CHmRun     263 non-null    int64
 10  CRuns      263 non-null    int64
 11  CRBI       263 non-null    int64
 12  CWalks     263 non-null    int64
 13  League     263 non-null    object
 14  Division   263 non-null    object
 15  PutOuts    263 non-null    int64
 16  Assists    263 non-null    int64
 17  Errors     263 non-null    int64
 18  Salary     263 non-null    float64
 19  NewLeague  263 non-null    object
```

```
dtypes: float64(1), int64(16), object(3)
memory usage: 43.1+ KB
```

Outlier analizi

[10]:
```
# Aykırı gözlemleri IQR yöntemiyle bulma ve temizleme
for col in numerik_degiskenler:
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1
    lower_limit = q1 - 1.5 * iqr
    upper_limit = q3 + 1.5 * iqr
    df = df[(df[col] >= lower_limit) & (df[col] <= upper_limit)]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 189 entries, 5 to 321
Data columns (total 20 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   AtBat      189 non-null    int64
 1   Hits       189 non-null    int64
 2   HmRun      189 non-null    int64
 3   Runs       189 non-null    int64
 4   RBI        189 non-null    int64
 5   Walks      189 non-null    int64
 6   Years      189 non-null    int64
 7   CAtBat     189 non-null    int64
 8   CHits      189 non-null    int64
 9   CHmRun     189 non-null    int64
 10  CRuns      189 non-null    int64
 11  CRBI       189 non-null    int64
 12  CWalks     189 non-null    int64
 13  League     189 non-null    object
 14  Division   189 non-null    object
 15  PutOuts    189 non-null    int64
 16  Assists    189 non-null    int64
 17  Errors     189 non-null    int64
 18  Salary     189 non-null    float64
 19  NewLeague  189 non-null    object
dtypes: float64(1), int64(16), object(3)
memory usage: 31.0+ KB
```

Yeni değişkenler oluşturma ve mevcut değişkenleri dönüştürme

[12]:
```
df['Years_group'] = pd.cut(df['Years'], bins=[0, 5, 10, 15, 20, 25],␣
 ↪labels=['0-5', '6-10', '11-15', '16-20', '21-25'])
```

[13]:
```
# Label encoding for binary categorical variables
binary_columns = ['League', 'Division', 'NewLeague']
le = LabelEncoder()
for col in binary_columns:
    df[col] = le.fit_transform(df[col])
```

[14]:
```
# One-hot encoding for multi-category categorical variables
df = pd.get_dummies(df, columns=['Years_group'], drop_first=True)
```

[15]:
```
df.head()
```

[15]:

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	…	\
5	594	169	4	74	51	35	11	4408	1133	19	…	
6	185	37	1	23	8	21	2	214	42	1	…	
7	298	73	0	24	24	7	3	509	108	0	…	
8	323	81	6	26	32	8	2	341	86	6	…	
10	574	159	21	107	75	59	10	4631	1300	90	…	

	Division	PutOuts	Assists	Errors	Salary	NewLeague	Years_group_6-10	\
5	1	282	421	25	750.000	0	False	
6	0	76	127	7	70.000	0	False	
7	1	121	283	9	100.000	0	False	
8	1	143	290	19	75.000	1	False	
10	0	238	445	22	517.143	0	True	

	Years_group_11-15	Years_group_16-20	Years_group_21-25
5	True	False	False
6	False	False	False
7	False	False	False
8	False	False	False
10	False	False	False

[5 rows x 24 columns]

standartlaştırma

[17]:
```
scaler = StandardScaler()
numerik_degiskenler = [col for col in numerik_degiskenler if col != 'Salary']
df[numerik_degiskenler] = scaler.fit_transform(df[numerik_degiskenler])
```

[18]:
```
df.head()
```

[18]:

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	\
5	1.419875	1.494453	-0.722329	0.914381	0.189057	-0.076355	1.481076	
6	-1.417930	-1.513746	-1.096357	-1.199221	-1.635723	-0.821872	-1.089649	
7	-0.633891	-0.693328	-1.221033	-1.157778	-0.956735	-1.567389	-0.804013	
8	-0.460431	-0.511013	-0.472977	-1.074892	-0.617241	-1.514137	-1.089649	

```
10  1.281107  1.266559  1.397163  2.282007  1.207539  1.201674  1.195440

      CAtBat     CHits    CHmRun  …  Division   PutOuts   Assists    Errors  \
5   1.874851  1.678548 -0.531325  …         1  0.672781  1.908541  2.423145
6  -1.170898 -1.158436 -1.069291  …         0 -1.142218  0.001613 -0.248125
7  -0.956664 -0.986813 -1.099178  …         1 -0.745738  1.013453  0.048683
8  -1.078669 -1.044021 -0.919856  …         1 -0.551903  1.058856  1.532721
10  2.036797  2.112807  1.590654  …         0  0.285112  2.064209  1.977933

     Salary  NewLeague  Years_group_6-10  Years_group_11-15  \
5   750.000          0             False               True
6    70.000          0             False              False
7   100.000          0             False              False
8    75.000          1             False              False
10  517.143          0              True              False

    Years_group_16-20  Years_group_21-25
5               False              False
6               False              False
7               False              False
8               False              False
10              False              False

[5 rows x 24 columns]
```

Modelleme

[20]:
```
# Veri ve hedef değişkenleri ayıralım
X = df.drop(columns=['Salary'])
y = df['Salary']
```

[21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[22]:
```
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "Gradient Boosting": GradientBoostingRegressor(),
    "AdaBoost": AdaBoostRegressor(),
    "XGBoost": XGBRegressor()
}
```

[23]:
```
performance_scores = {}
```

[24]:
```
for model_name, model in models.items():
    model.fit(X_train, y_train)
```

```
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)
    performance_scores[model_name] = {'RMSE': rmse, 'R2': r2}
```

[25]:
```
# Performans skorlarını inceleyelim
for model_name, scores in performance_scores.items():
    print(f"{model_name} - RMSE: {scores['RMSE']:.4f}, R2: {scores['R2']:.4f}")
```

```
Linear Regression - RMSE: 186.0919, R2: 0.5880
Decision Tree - RMSE: 207.1829, R2: 0.4893
Random Forest - RMSE: 132.8424, R2: 0.7901
Gradient Boosting - RMSE: 133.1862, R2: 0.7890
AdaBoost - RMSE: 139.1093, R2: 0.7698
XGBoost - RMSE: 119.5009, R2: 0.8301
```

[26]:
```
# En iyi 4 modeli seçelim
top_4_models = sorted(performance_scores, key=lambda x:␣
 ↪performance_scores[x]['R2'], reverse=True)[:4]
print("En iyi 4 model:", top_4_models)
```

```
En iyi 4 model: ['XGBoost', 'Random Forest', 'Gradient Boosting', 'AdaBoost']
```

Hiperparametre Optimizasyonu

[28]:
```
param_grid = {
    "Random Forest": {
        'n_estimators': [100, 200, 500],
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10]
    },
    "Gradient Boosting": {
        'n_estimators': [100, 200],
        'learning_rate': [0.01, 0.1, 0.2],
        'max_depth': [3, 4, 5]
    },
    "AdaBoost": {
        'n_estimators': [50, 100, 200],
        'learning_rate': [0.01, 0.1, 0.5]
    },
    "XGBoost": {
        'n_estimators': [100, 200],
        'learning_rate': [0.01, 0.1, 0.2],
        'max_depth': [3, 4, 5]
    }
}
```

[29]:
```
best_params = {}
best_scores = {}
```

[30]:
```
# Hiperparametre optimizasyonunu gerçekleştirelim
for model_name in top_4_models:
    grid_search = GridSearchCV(models[model_name], param_grid[model_name],␣
 ↪cv=5, scoring='r2', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    best_params[model_name] = grid_search.best_params_
    best_scores[model_name] = grid_search.best_score_
```

[31]:
```
# En iyi hiperparametreler ve doğruluk skorları
for model_name in best_params:
    print(f"{model_name} - Best Params: {best_params[model_name]} - Best Score:␣
 ↪{best_scores[model_name]:.4f}")
```

```
XGBoost - Best Params: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators':
200} - Best Score: 0.5804
Random Forest - Best Params: {'max_depth': 20, 'min_samples_split': 10,
'n_estimators': 100} - Best Score: 0.5616
Gradient Boosting - Best Params: {'learning_rate': 0.01, 'max_depth': 4,
'n_estimators': 200} - Best Score: 0.5698
AdaBoost - Best Params: {'learning_rate': 0.1, 'n_estimators': 200} - Best
Score: 0.6034
```

[32]:
```
# Seçilen modelleri en iyi hiperparametreler ile tekrar eğitelim
final_models = {}
for model_name in best_params:
    model = models[model_name].set_params(**best_params[model_name])
    model.fit(X_train, y_train)
    final_models[model_name] = model
```

[33]:
```
# Final modellerin performanslarını test seti üzerinde inceleyelim
final_performance_scores = {}
for model_name, model in final_models.items():
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)
    final_performance_scores[model_name] = {'RMSE': rmse, 'R2': r2}
```

[34]:
```
# Final performans skorları
for model_name, scores in final_performance_scores.items():
    print(f"{model_name} Final RMSE: {scores['RMSE']:.4f}, Final R2:␣
 ↪{scores['R2']:.4f}")
```

```
XGBoost Final RMSE: 125.2226, Final R2: 0.8135
Random Forest Final RMSE: 134.7161, Final R2: 0.7841
```

```
Gradient Boosting Final RMSE: 133.6192, Final R2: 0.7876
AdaBoost Final RMSE: 132.8920, Final R2: 0.7899
```

[36]:
```
# en iyi model olan xgboost u export etme
model_filename = 'xgboost_model.pkl'
joblib.dump(final_models['XGBoost'], model_filename)
```

[36]:
```
['xgboost_model.pkl']
```