Lineer Regresyon ile Hitters Salary Prediction Çalışması

Fotoğraf: www.parkgrandlondon.co.uk

Özet

Bu çalışmada, maaş bilgileri ve 1986 yılına ait kariyer istatistikleri paylaşılan beyzbol oyuncularının verileri Lineer Regresyon tekniği kullanılarak maaş tahminlemesi yapılmıştır. Çalışma süresince, oyuncu maaş değişkeninde etkili olan tüm parametreler veri seti çerçevesinde ele alınmış olup, anlamlılıkları incelenmiştir. Çalışma sonunda elde edilen model sayesinde, belirli parametreleri verilen bir oyuncunun maaş tahmini %75 başarı oranında tahmin edilebilmiştir. Model öncesinde, veri setinin doğru bir şekilde yorumlanabilmesi ve işlenebilmesi için izlenen işlem basamakları sırasıyla şu şekildedir;

- *Exploratory Data Analysis (Keşifçi Veri Analizi)*
- *Korelasyon Analizi*
- *Feature Engineering (Özellik Mühendisliği)*
- *Feature Extraction (Değişken Türetme)*
- *Encoding (Kodlama)*
- *Model Oluşturma*
- *Tahminleme*
- *Tahminleme Başarı Testi*

Veri Seti

https://www.kaggle.com/floser/hitters

Bu veri seti orijinal olarak Carnegie Mellon Üniversitesi'nde bulunan StatLib kütüphanesinden alınmıştır. Veri seti 1988 ASA Grafik Bölümü Poster Oturumu'nda kullanılan verilerin bir parçasıdır. Maaş verileri orijinal olarak Sports Illustrated, 20 Nisan 1987'den alınmıştır. 1986 ve kariyer istatistikleri, Collier Books, Macmillan Publishing Company, New York tarafından yayınlanan 1987 Beyzbol Ansiklopedisi Güncellemesinden elde edilmiştir. Şuanda da Kaggle üzerinden paylaşılmaktadır.

Toplamda 20 Değişken, 6440 Gözlemden oluşmakta olup 20.91 KB boyutundadır.

Veri seti üzerindeki birakım değişkenler sırasıyla şöyledir;

***AtBat:** 1986-1987 sezonunda bir beyzbol sopası ile topa yapılan vuruş sayısı*

***Hits:** 1986-1987 sezonundaki isabet sayısı*

***HmRun:** 1986-1987 sezonundaki en değerli vuruş sayısı*

***Runs:** 1986-1987 sezonunda takımına kazandırdığı sayı*

***Walks:** Karşı oyuncuya yaptırılan hata sayısı*

***Years:** Oyuncunun major liginde oynama süresi (sene)*

***CAtBat:** Oyuncunun kariyeri boyunca topa vurma sayısı*

***CHits:** Oyuncunun kariyeri boyunca yaptığı isabetli vuruş sayısı*

***CRuns:** Oyuncunun kariyeri boyunca takımına kazandırdığı sayı*

***CRBI:** Oyuncunun kariyeri boyunca koşu yaptırdırdığı oyuncu sayısı*

***PutOuts:** Oyun icinde takım arkadaşınla yardımlaşma*

***Assits:** 1986-1987 sezonunda oyuncunun yaptığı asist sayısı*

***Errors:** 1986-1987 sezonundaki oyuncunun hata sayısı*

***Salary:** Oyuncunun 1986-1987 sezonunda aldığı maaş(bin uzerinden)*

***NewLeague:** 1987 sezonunun başında oyuncunun ligini gösteren A ve N seviyelerine sahip bir faktör*

1- Exploratory Data Analysis (Keşifçi Veri Analizi)

1.1-Veri Setini İnceleme

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import missingno as msno
from datetime import date
from sklearn.metrics import accuracy_score
from sklearn.neighbors import LocalOutlierFactor
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, StandardScaler, RobustScaler
from catboost import CatBoostClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,roc_auc_score
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split, cross_val_score, cross_validate
from sklearn.linear_model import LinearRegression
from dython import nominal
from dython.data_utils import split_hist
from IPython.core.display import HTML
from scipy.stats import norm
```

In [2]:
```
pd.set_option("display.max_columns", 8)
pd.set_option("display.max_rows", None)
pd.set_option("display.float_format", lambda x: "%.3f" % x)
pd.set_option("display.width", 500)
```

In [3]:
```
def load_hitters():
    df = pd.read_csv("C:/Users/eyp_d/Desktop/VBO-Bootcamp-Dersler/7.Hafta/Ders Öncesi Notlar/datas
    return df
```

In [4]:
```
df = load_hitters()
df.head()
```

Out[4]:

	AtBat	Hits	HmRun	Runs	...	Assists	Errors	Salary	NewLeague
0	293	66	1	30	...	33	20	NaN	A
1	315	81	7	24	...	43	10	475.000	N
2	479	130	18	66	...	82	14	480.000	A
3	496	141	20	65	...	11	3	500.000	N
4	321	87	10	39	...	40	4	91.500	N

5 rows × 20 columns

In [5]:
```
df.shape  # Veri seti boyutu
```

Out[5]:
```
(322, 20)
```

In [6]:
```
df.describe() # Veri seti değişkenlerine ait özet istatistik bilgisi
```

Out[6]:

	AtBat	Hits	HmRun	Runs	...	PutOuts	Assists	Errors	Salary
count	322.000	322.000	322.000	322.000	...	322.000	322.000	322.000	263.000
mean	380.929	101.025	10.770	50.910	...	288.938	106.913	8.040	535.926
std	153.405	46.455	8.709	26.024	...	280.705	136.855	6.368	451.119
min	16.000	1.000	0.000	0.000	...	0.000	0.000	0.000	67.500
25%	255.250	64.000	4.000	30.250	...	109.250	7.000	3.000	190.000
50%	379.500	96.000	8.000	48.000	...	212.000	39.500	6.000	425.000
75%	512.000	137.000	16.000	69.000	...	325.000	166.000	11.000	750.000
max	687.000	238.000	40.000	130.000	...	1378.000	492.000	32.000	2460.000

8 rows × 17 columns

CatBat, CWalks, PutOuts gibi değişkenlerde outlier(aykırı değer) olabilir.

In [7]:
```
df.columns = [col.upper() for col in df.columns]
df.head(2)
```

Out[7]:

	ATBAT	HITS	HMRUN	RUNS	...	ASSISTS	ERRORS	SALARY	NEWLEAGUE
0	293	66	1	30	...	33	20	NaN	A
1	315	81	7	24	...	43	10	475.000	N

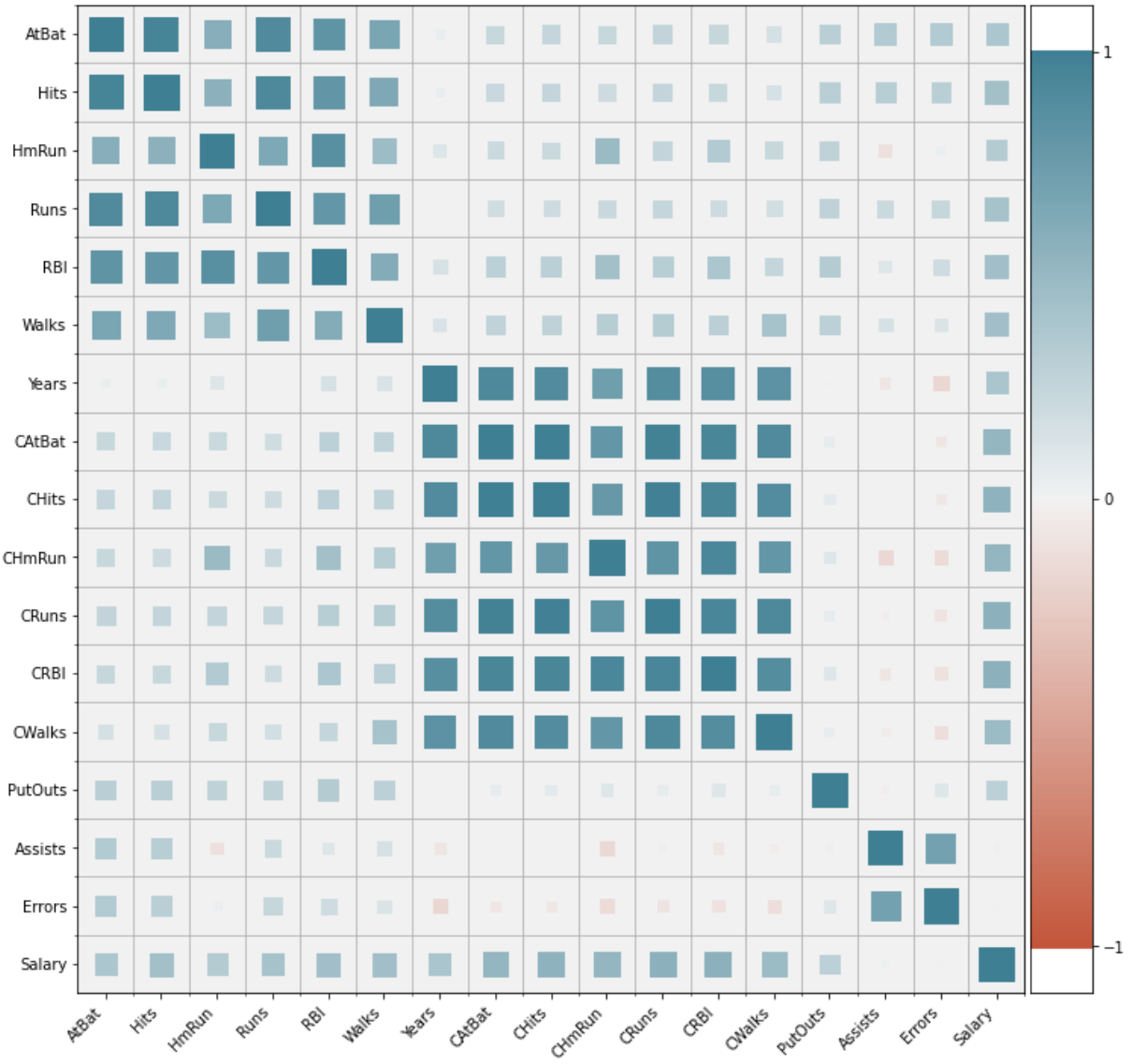
2 rows × 20 columns

Değişken adları kullanım kolaylığı için büyük harflere çevirildi.

1.2-Değişkenler Arasındaki Korelasyon Analizi

Tüm değişkenlerin birbiri arasındaki korelasyonunu incelemek için ana dataframe'i üzerinde değişiklik yapmadan Label(Binary) Encoding işlemi yapılacaktır.

In [9]:
```
corr_df = load_hitters()  # Korelasyon grafiği kaynak kodları için: https://www.kaggle.com/drazen/
corr = corr_df.corr()
plt.figure(figsize=(12, 12))
corrplot(corr)
```



Görüldüğü üzere bağımlı değişken(SALARY) ile en yüksek korelasyona sahip değişkenler: HITS, RBI, WALKS, CRUNS, RUNS ... olarak sıralanabilir.

1.3-Numerik ve Kategorik Değişkenlerin Analizi

grab_col_names() fonksiyonu bir dataframedeki değişkenlerin türlerini göstermek için kullanılır.

In [10]:
```
def grab_col_names(dataframe, cat_th=10, car_th=20):

    # cat_cols, cat_but_car
    cat_cols = [col for col in dataframe.columns if dataframe[col].dtypes == "O"]
    num_but_cat = [col for col in dataframe.columns if dataframe[col].nunique() < cat_th and
                   dataframe[col].dtypes != "O"]
    cat_but_car = [col for col in dataframe.columns if dataframe[col].nunique() > car_th and
                   dataframe[col].dtypes == "O"]
    cat_cols = cat_cols + num_but_cat
    cat_cols = [col for col in cat_cols if col not in cat_but_car]

    # num_cols
    num_cols = [col for col in dataframe.columns if dataframe[col].dtypes != "O"]
    num_cols = [col for col in num_cols if col not in num_but_cat]

    return cat_cols, num_cols, cat_but_car
```

In [11]:
```
cat_cols, num_cols, cat_but_car = grab_col_names(df)
```

1.3.1-Kategorik Değişkenlerin Bağımlı Değişkene Göre Analizi

Veri setindeki değişkenlerin bağımlı değişkenle olan korelasyon grafiği incelenerek, yüksek korelasyon değerlerine sahip kategorik değişkenler bu bölümde incelenecektir.

cat_summary() veri setindeki kategorik değişkenlerin dağılımını veren özet fonksiyonudur.

In [12]:
```
def cat_summary(dataframe, col_name, target_col):
    print(pd.DataFrame({col_name: dataframe[col_name].value_counts(),
                        "Ratio": 100 * dataframe[col_name].value_counts() / len(dataframe),
                        f"{target_col}"+"_Mean": df.groupby(col_name)[target_col].mean()}))
```

cat_distribution_plot() veri setindeki kategorik değişkenlerin bağımlı değişkene göre dağılımını gösteren grafik fonksiyonudur.

In [13]:
```
def cat_distribution_plot(df, cat_col, target_col):
    split_hist(df, target_col, split_by=cat_col, ylabel="Gözlem Sayısı",
               title=f"{target_col}'e göre {cat_col} Dağılımı", bins=25, figsize=(12.1,4))
    plt.show()
```

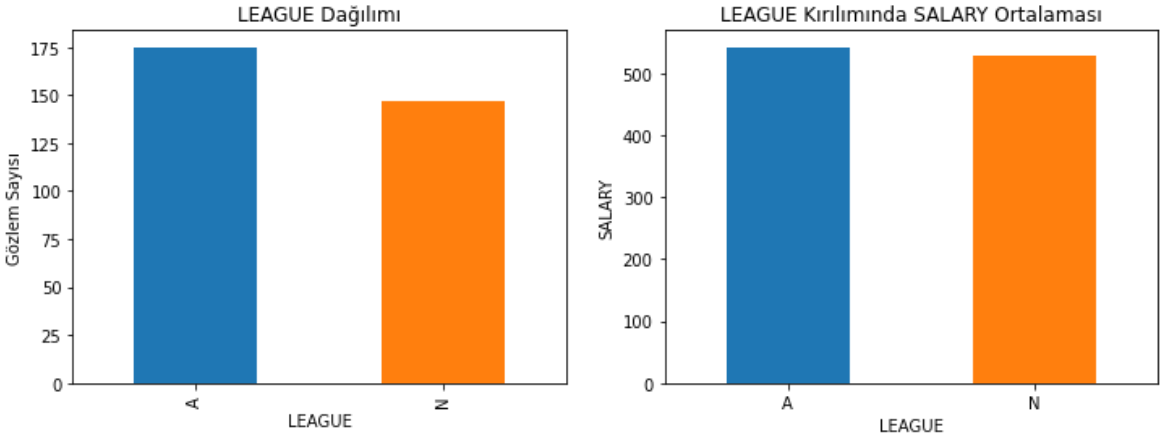
cat_analyser_plot() veri setindeki kategorik değişkenlerin dağılımı ve bağımlı değişkenle olan ilişkisini gösteren grafik fonksiyonudur.

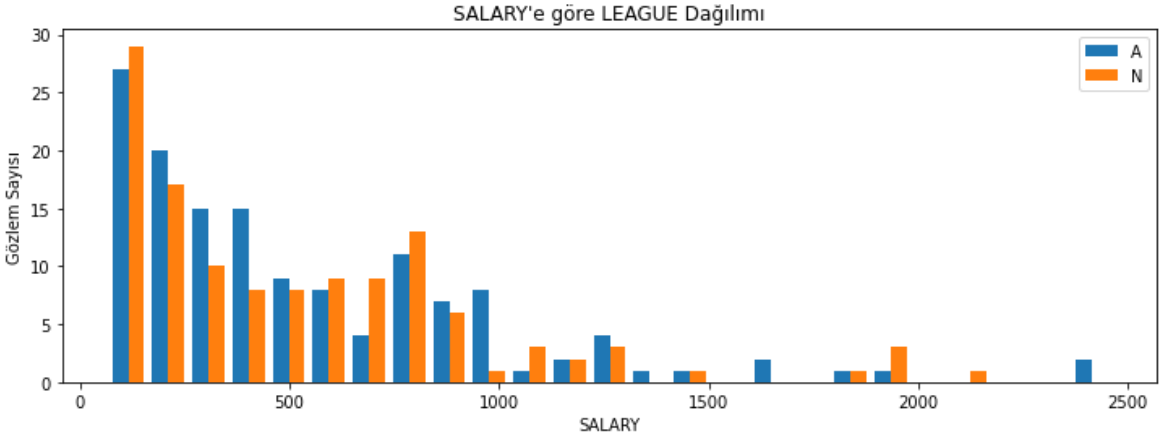
In [14]:
```
def cat_analyser_plot(df, cat_col, target_col):
    fig, axes = plt.subplots(1, 2, figsize=(12, 4))

    df[cat_col].value_counts().plot(kind='bar', ax=axes[0], color=['#1f77b4', '#ff7f0e', '#2ca02c
    axes[0].set_title(f"{cat_col} Dağılımı")
    axes[0].set_xlabel(f"{cat_col}")
    axes[0].set_ylabel("Gözlem Sayısı")

    df.groupby(cat_col)[target_col].mean().plot(kind='bar', ax=axes[1], color=['#1f77b4', '#ff7f0
    axes[1].set_title(f"{cat_col}"+" "+"Kırılımında"+" "+f"{target_col}"+" "+ "Ortalaması")
    axes[1].set_xlabel(f"{cat_col}")
    axes[1].set_ylabel(f"{target_col}")
    plt.xticks(rotation=0)
    plt.show()

    cat_distribution_plot(df, cat_col, target_col)
```

In [15]:
```
cat_summary(df, "LEAGUE", "SALARY")
```

```
   LEAGUE  Ratio  SALARY_Mean
A     175 54.348      542.000
N     147 45.652      529.118
```

In [16]:
```
cat_analyser_plot(df, "LEAGUE", "SALARY")
```



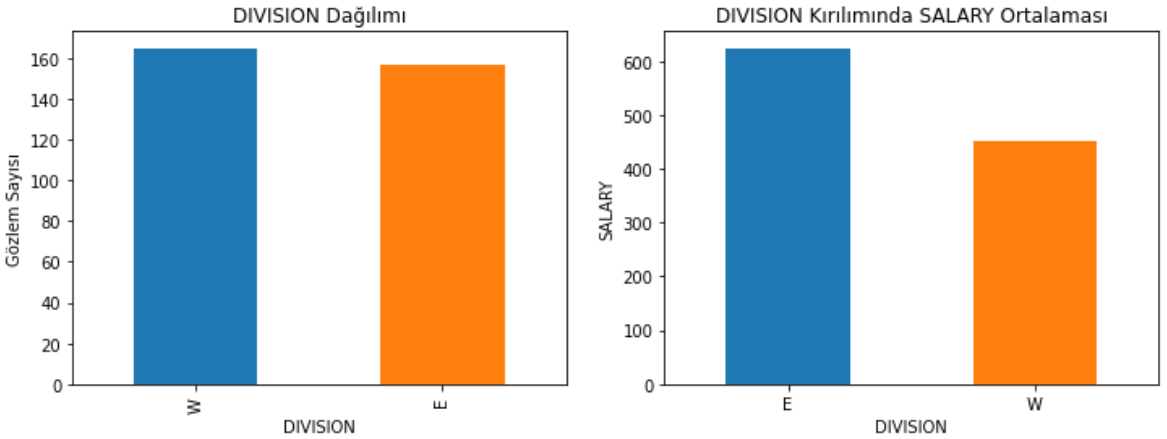


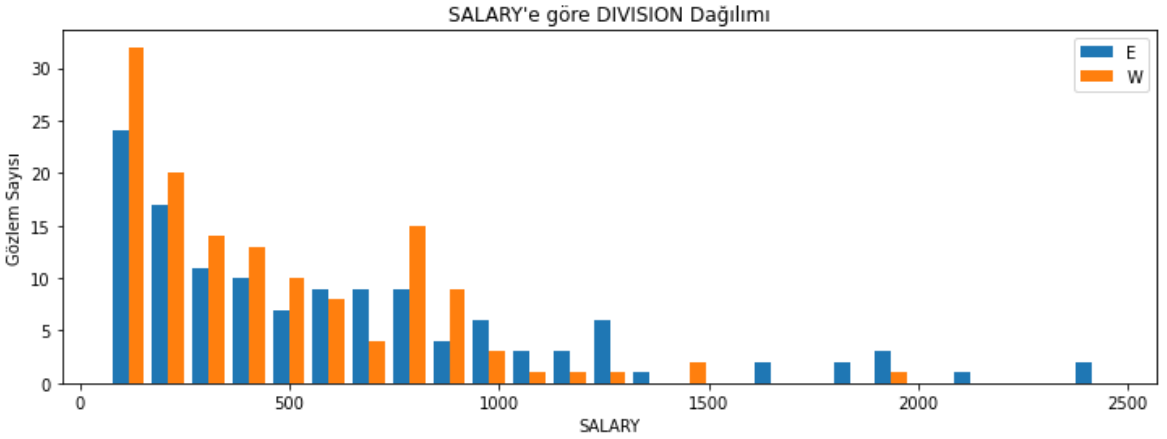
Yukarıdaki grafiklerden anlaşılacağı üzere, N liginde oynayan oyuncular A liginde oynayanlara göre daha az olmasına karşın SALARY ortalamasında yaklaşık olarak benzer sonuç göstermişler. Buna göre, bir oyuncu için N liginde oynamanın daha fazla maaş(SALARY) getirdiği yorumu yapılabilir.

In [17]:
```
cat_summary(df, "DIVISION", "SALARY")
```

```
   DIVISION  Ratio  SALARY_Mean
E       157 48.758      624.271
W       165 51.242      450.877
```

In [18]:
```
cat_analyser_plot(df, "DIVISION", "SALARY")
```



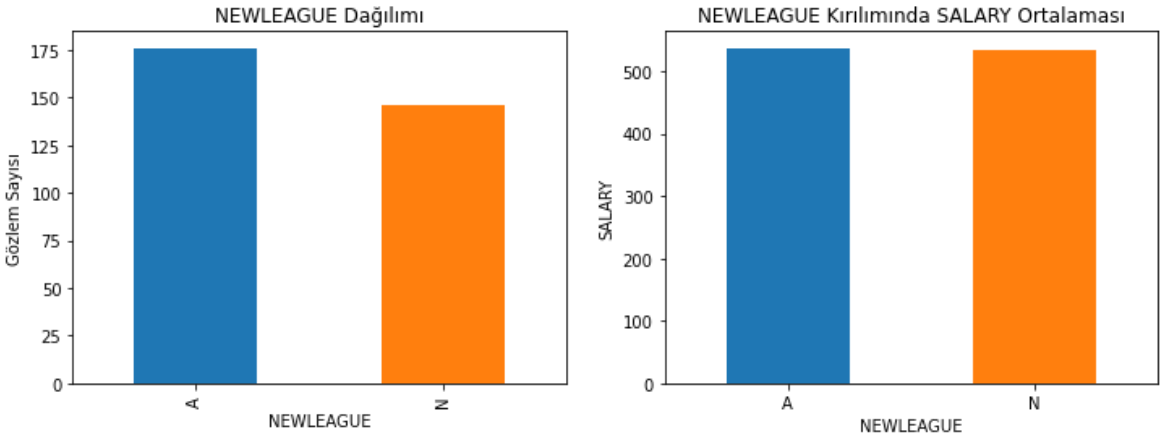


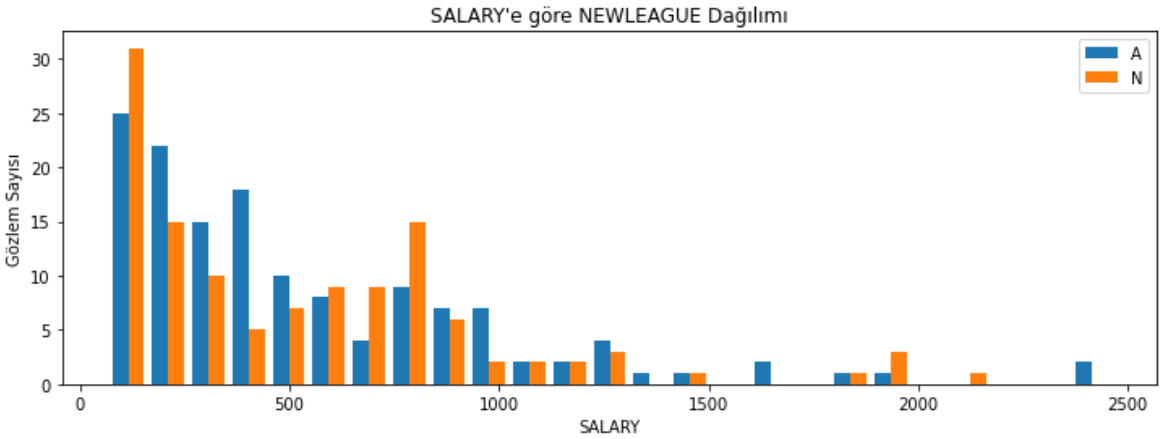
Oyuncuların oyun sahasındaki pozisyonların dağılımı neredeyse aynı sonuç gösterse de E pozisyonunda oynayan oyuncuların maaş(SALARY) ortalaması daha yüksek. Bu sebep, oyuncuya ait SALARY değişkeni oynadığı pozisyona göre değişmekte ve E pozisyonu için pozitif yönlüdür.

In [19]:
```
cat_summary(df, "NEWLEAGUE", "SALARY")
```

```
   NEWLEAGUE  Ratio  SALARY_Mean
A        176 54.658      537.113
N        146 45.342      534.554
```

In [20]:
```
cat_analyser_plot(df, "NEWLEAGUE", "SALARY")
```





NEWLEAGUE değişkeninin dağılımı da beklenildiği üzere LEAGUE değişken dağılımına benzemektedir. Aynı yorumlar bu değişken için de yapılabilir.

1.3.2-Nümerik Değişkenlerin Bağımlı Değişkene Göre Analizi

Veri setindeki değişkenlerin bağımlı değişkenle olan korelasyon grafiği incelenerek, yüksek korelasyon değerlerine sahip nümerik değişkenler bu bölümde incelenecektir.

In [21]:
```
num_cols  # Nümerik değişkenler
```

Out[21]:
```
['ATBAT',
 'HITS',
 'HMRUN',
 'RUNS',
 'RBI',
 'WALKS',
 'YEARS',
 'CATBAT',
 'CHITS',
 'CHMRUN',
 'CRUNS',
 'CRBI',
 'CWALKS',
 'PUTOUTS',
 'ASSISTS',
 'ERRORS',
 'SALARY']
```

num_summary() veri setindeki nümerik değişkenlerin dağılımını veren özet fonksiyonudur.

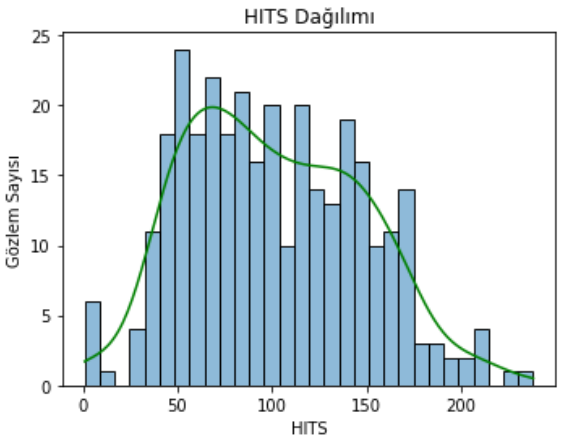
In [22]:
```
def num_summary(dataframe, numerical_col, plot=False):
    quantiles = [0.05, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 0.95, 0.99]
    print("\t\t\t"+f"{numerical_col}"+" Nümerik Değişken Özet İstatistiği")
    print("\t\t\t-----------------------------------------------")
    print(pd.DataFrame(dataframe[numerical_col].describe(quantiles)).T, "\n")

    if plot:
        dataframe[numerical_col].hist(bins=20)
        plt.xlabel(numerical_col)
        plt.title(numerical_col)
        plt.show()
```

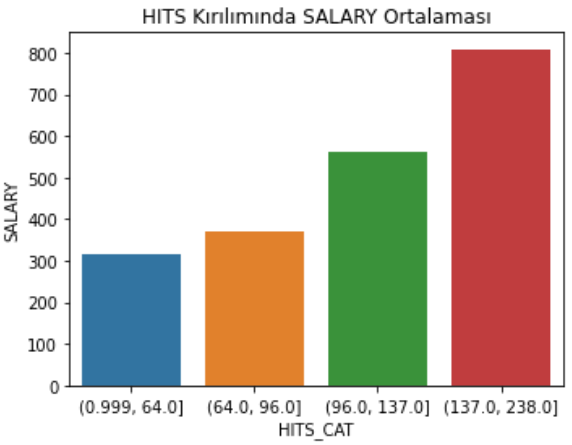
num_analyser_plot() veri setindeki nümerik değişkenlerin dağılımı ve bağımlı değişkenle olan ilişkisini gösteren grafik fonksiyonudur.

In [23]:
```
def num_analyser_plot(df, num_col, target_col):
    fig, axes = plt.subplots(1, 2, figsize=(12, 4))

    sns.histplot(df[num_col], kde=True, bins=30, ax=axes[0]);
    axes[0].lines[0].set_color('green')
    axes[0].set_title(f"{num_col}"+" "+"Dağılımı")
    axes[0].set_ylabel("Gözlem Sayısı")

    quantiles = [0, 0.25, 0.50, 0.75, 1]
    num_df = df.copy()
    num_df[f"{num_col}"+"_CAT"] = pd.qcut(df[num_col], q=quantiles) # nümerik değişken kategorize
    df_2 = num_df.groupby(f"{num_col}"+"_CAT")[target_col].mean()

    sns.barplot(x=df_2.index, y=df_2.values);
    axes[1].set_title(f"{num_col} Kırılımında {target_col} Ortalaması")
    axes[1].set_ylabel(f"{target_col}")

    plt.show()
```

In [24]:
```
num_summary(df, "HITS")
num_analyser_plot(df, "HITS", "SALARY")
```

```
                        HITS Nümerik Değişken Özet İstatistiği
                        -----------------------------------------------
       count    mean     std   min  ...     90%     95%     99%     max
HITS 322.000 101.025 46.455 1.000  ... 163.000 174.000 210.790 238.000

[1 rows x 17 columns]
```

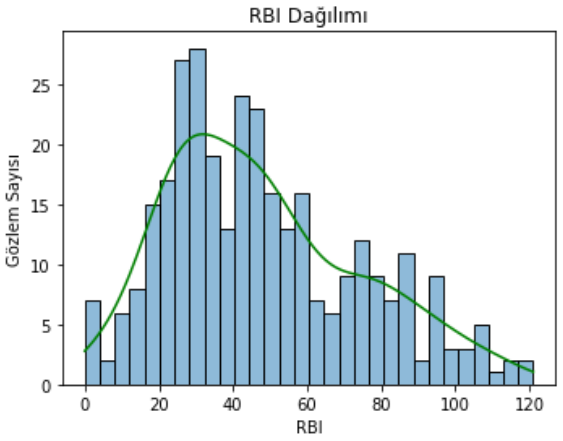


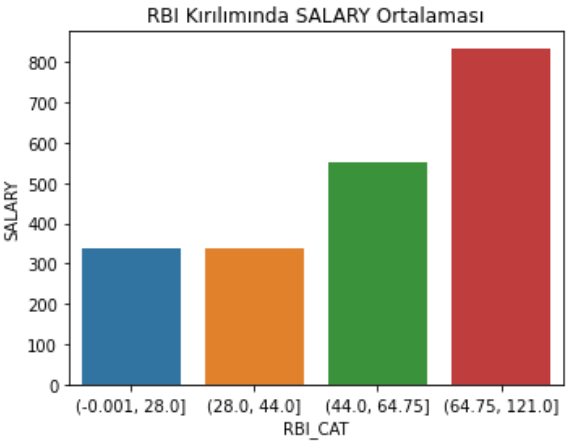


Yukarıdaki grafiğe göre, bir oyuncu için HITS(1986-1987 sezonundaki isabet sayısı) arttıkça ortalama SALARY değeri artmaktadır.

In [25]:
```
num_summary(df, "RBI")
num_analyser_plot(df, "RBI", "SALARY")
```

```
                    RBI Nümerik Değişken Özet İstatistiği
                    --------------------------------------------
      count   mean    std   min  ...    90%    95%     99%     max
RBI 322.000 48.028 26.167 0.000  ... 86.000 96.000 112.370 121.000

[1 rows x 17 columns]
```

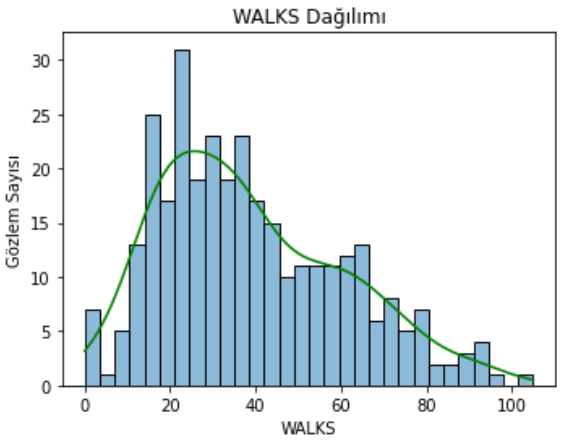


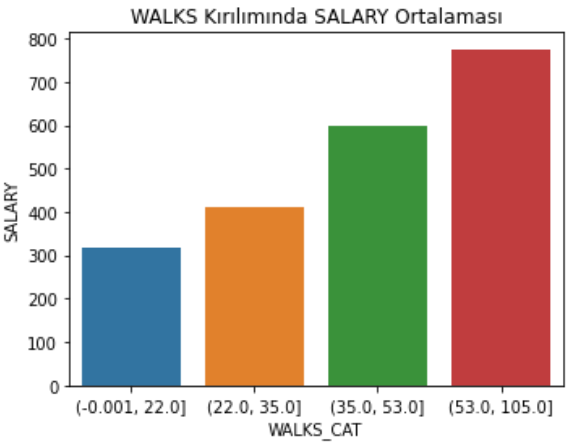


Yukarıdaki grafiğe göre, bir oyuncu için RBI (Bir vurucunun vuruş yaptığında koşu yaptırdığı oyuncu sayısı) arttıkça ortalama SALARY değeri artmaktadır.

In [26]:
```
num_summary(df, "WALKS")
num_analyser_plot(df, "WALKS", "SALARY")
```

```
                    WALKS Nümerik Değişken Özet İstatistiği
                    --------------------------------------------
       count   mean    std   min  ...    90%    95%    99%     max
WALKS 322.000 38.742 21.639 0.000  ... 69.900 78.000 93.580 105.000

[1 rows x 17 columns]
```

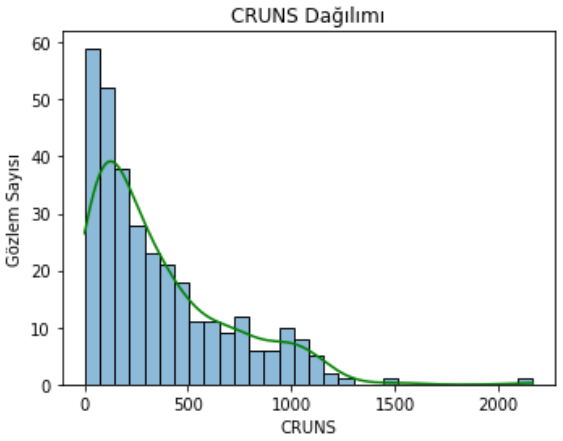


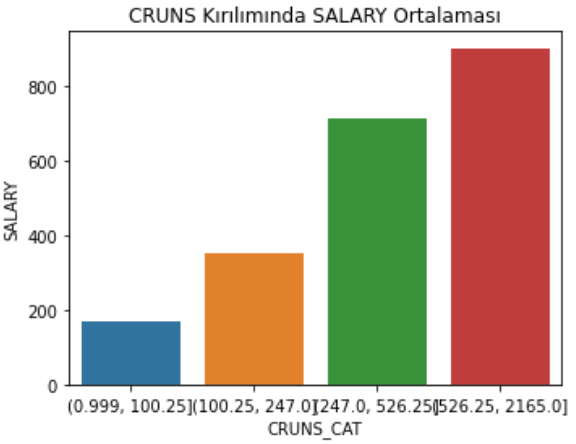


Yukarıdaki grafiğe göre, bir oyuncu için WALKS (karşı oyuncuya yaptırılan hata sayısı) arttıkça ortalama SALARY değeri artmaktadır.

In [27]:
```
num_summary(df, "CRUNS")
num_analyser_plot(df, "CRUNS", "SALARY")
```

```
                      CRUNS Nümerik Değişken Özet İstatistiği
                      -----------------------------------------
        count     mean      std   min  ...      90%       95%       99%       max
CRUNS 322.000 358.795 334.106 1.000  ... 895.700 1032.300 1174.370 2165.000

[1 rows x 17 columns]
```





Ykarıdaki grafiklerden, CRUNS(Oyuncunun kariyeri boyunca takımına kazandırdığı sayı) değişkeninin veri seti üzerinde 0-150 değerleri arasında kaldığı ve bu sebeple SALARY değişkenine doğrudan pozitif yönlü etki ettiğini görmek mümkün.

1.4-Aykırı Gözlem Analizi

Bu bölümde veri seti değişkenlerine ait verilerdeki aykırı gözlemler tespit edilecektir.

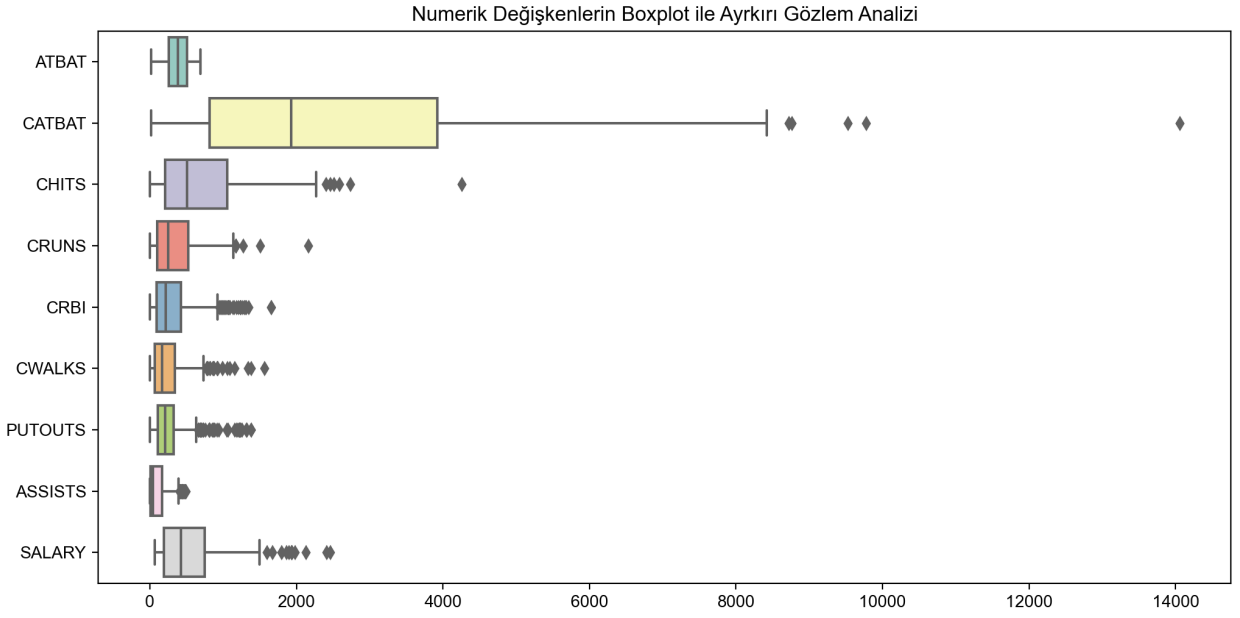
outliers_boxplot() Veri setindeki nümerik değişkenlerdeki aykırı gözlemleri gösteren grafik fonksiyonudur.

In [28]:
```
def outliers_boxplot(dataframe, num_cols):
    plt.figure(figsize=(12,6),dpi=200)
    plt.title("Numerik Değişkenlerin Boxplot ile Ayrkırı Gözlem Analizi")
    sns.set_theme(style="whitegrid")
    sns.boxplot(data=df.loc[:, num_cols], orient="h", palette="Set3")
    plt.show()
```

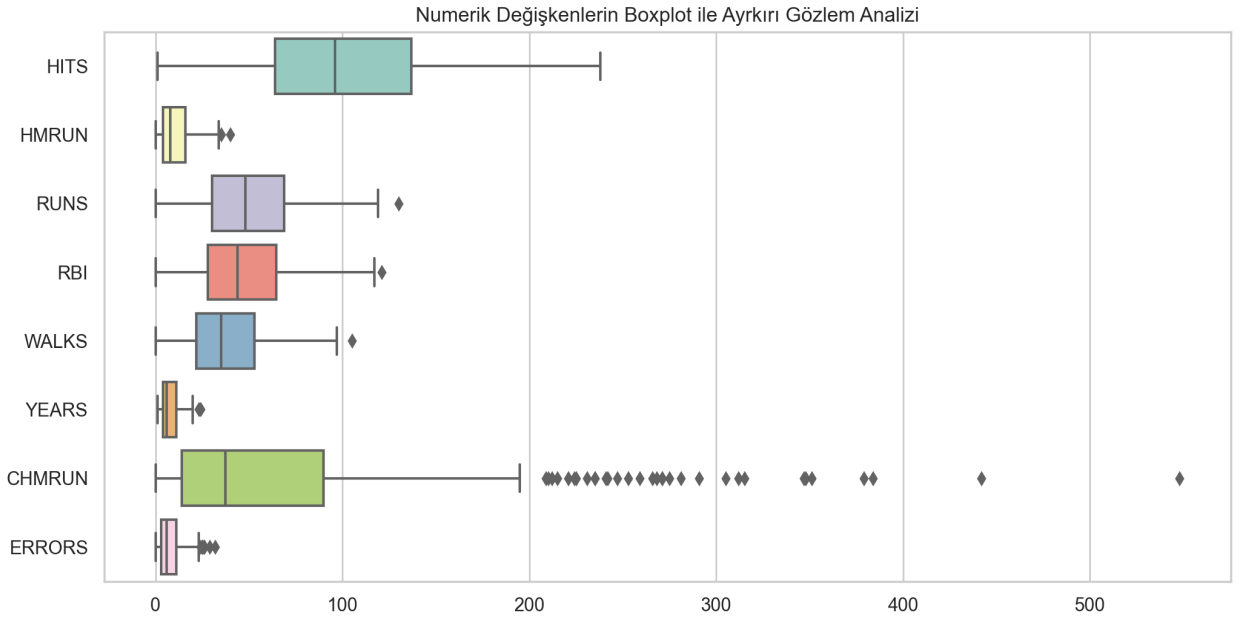
Outlier bulunduran değişkenler çok olduğundan grafik üzerinde net incelenebilmesi adına ikiye ayrılmıştır.

In [29]:
```
df_mean = int(df.mean(numeric_only=True).quantile(0.5))
high_cols = []
low_cols = []
for col in num_cols:
    if df[col].mean() >= df_mean:
        high_cols.append(col)
    else:
        low_cols.append(col)
```

In [30]:
```
outliers_boxplot(df, high_cols)  # Yüksek değerlikli nümerik değişkenlerin aykırı gözlem analizi
```



In [31]:
```
outliers_boxplot(df, low_cols)  # Düşük değerlikli nümerik değişkenlerin aykırı gözlem analizi
```



Görüldüğü gibi birçok nümerik değişkende outliers görülmektedir.

Sayısal olarak inceleme

outlier_thresholds(), değişkenin outlier threshold değerlerini hesaplayan fonksiyondur

In [32]:
```
def outlier_thresholds(dataframe, col_name, q1=0.25, q3=0.75):
    quartile1 = dataframe[col_name].quantile(q1)
    quartile3 = dataframe[col_name].quantile(q3)
    interquantile_range = quartile3 - quartile1
    up_limit = quartile3 + 1.5 * interquantile_range
    low_limit = quartile1 - 1.5 * interquantile_range
    return low_limit, up_limit
```

check_outlier(), değişkende outlier olup olmadığını dönen fonksiyondur

In [33]:
```
def check_outlier(dataframe, col_name):
    low_limit, up_limit = outlier_thresholds(dataframe, col_name)
    if dataframe[(dataframe[col_name] > up_limit) | (dataframe[col_name] < low_limit)].any(axis=No
        return True
    else:
        return False
```

In [34]:
```
outliers_list = []
for col in num_cols:
    check_return = check_outlier(df, col)
    print(f"{col}:", check_return)
    if check_return:
        outliers_list.append(col)   # Outlier değişkenler liste olarak tutulur.
```

```
ATBAT: False
HITS: False
HMRUN: True
RUNS: True
RBI: True
WALKS: True
YEARS: True
CATBAT: True
CHITS: True
CHMRUN: True
CRUNS: True
CRBI: True
CWALKS: True
PUTOUTS: True
ASSISTS: True
ERRORS: True
SALARY: True
```

Outlier gözlem bulunduran tüm değişkenlere True olarak görülebilir.

1.5-Eksik Gözlem Analizi

In [35]:
```
df.isnull().values.any() # Veri setinde eksik gözlem(missing value) var mı?
```

Out[35]:
```
True
```

Missing value gözlem değerlerine sahip değişkenler var.

missing_values_table(), missing value gözlemlere sahip değişkenleri, missing value sayısını ve ortalamasını dönen fonksiyondur.

In [36]:
```
def missing_values_table(dataframe, na_name=False):
    na_columns = [col for col in dataframe.columns if dataframe[col].isnull().sum() > 0]
    n_miss = dataframe[na_columns].isnull().sum().sort_values(ascending=False)
    ratio = (dataframe[na_columns].isnull().sum() / dataframe.shape[0] * 100).sort_values(ascendi
    missing_df = pd.concat([n_miss, np.round(ratio, 2)], axis=1, keys=['n_miss', 'ratio'])
    print(missing_df, end="\n")
    if na_name:
        return na_columns
```

In [37]:
```
missing_values_table(df, na_name=True)
```

```
        n_miss  ratio
SALARY      59 18.320
```

Out[37]:
```
['SALARY']
```

Bağımlı değişkende missing value değerleri görülmekte.

2-Feature Engineering (Özellik Mühendisliği)

Buraya kadarki kısım, Veri setine ilk bakışla ilgiliydi. EDA olarak da tanımlanan buraya kadarki kısımda veri üzerinde herhangi bir değişiklik yapılmadı.

Feature Engineering bölümüyle ile birlikte veri seti, geliştirilecek model için sayısallaştırılacaktır.

2.1-Eksik ve Aykırı Değerleri Düzeltme

Bu bölümde, veri setinde eksik gözlemlere sahip değişkenler baskılama yöntemiyle düzeltiliyor olacak.

2.1.1-Aykırı Değerlerin Düzeltilmesi

In [38]:
```
outliers_list  # Aykırı değer gözlemler barındıran değişkenlerin listesi
```

Out[38]:
```
['HMRUN',
 'RUNS',
 'RBI',
```

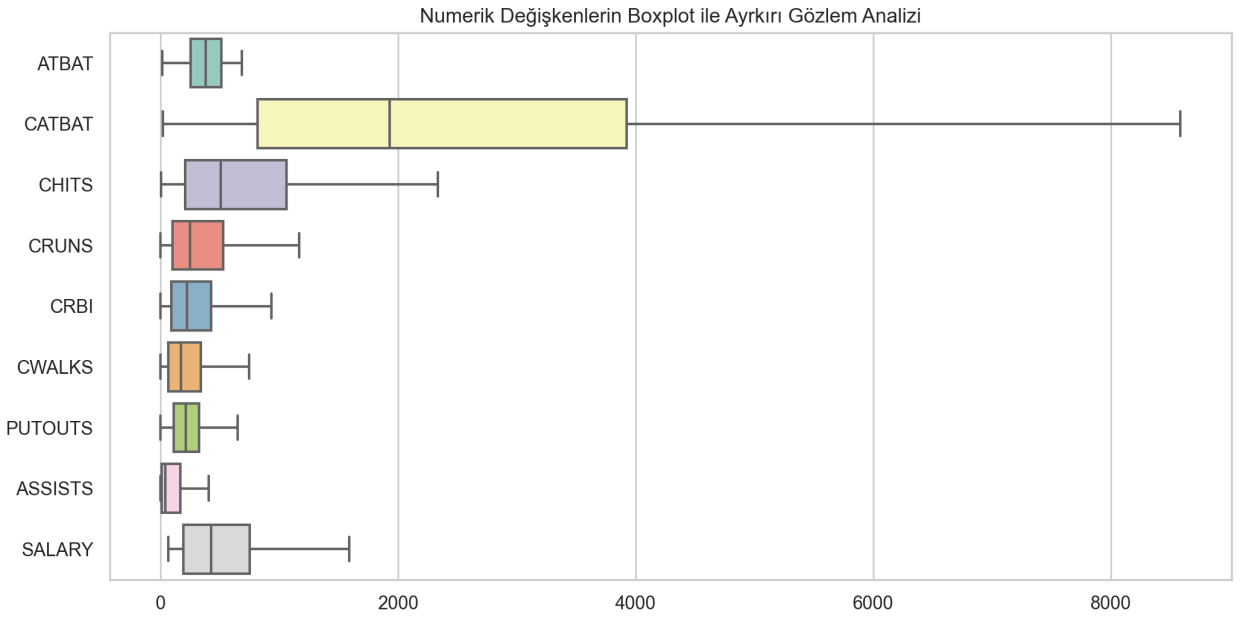
```
'WALKS',
'YEARS',
'CATBAT',
'CHITS',
'CHMRUN',
'CRUNS',
'CRBI',
'CWALKS',
'PUTOUTS',
'ASSISTS',
'ERRORS',
'SALARY']
```

replace_with_thresholds() veri setindeki outlier değerleri belirlenen threshold değerine göre baskılayan fonksiyondur.

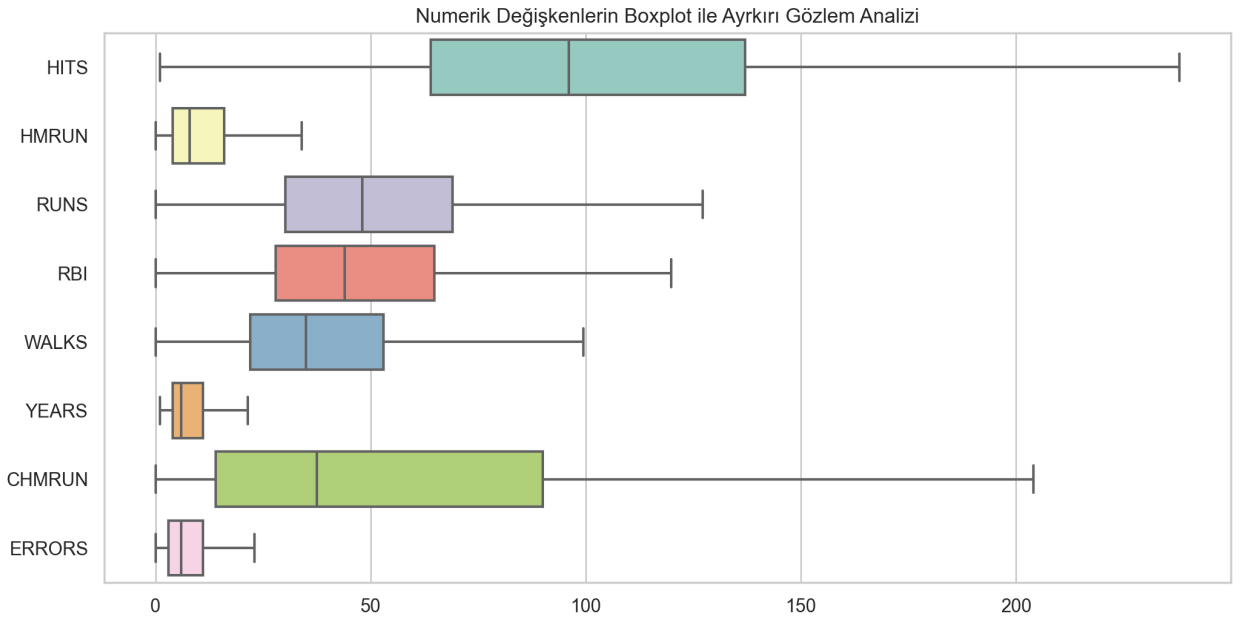
In [39]:
```
def replace_with_thresholds(dataframe, variable):
    low_limit, up_limit = outlier_thresholds(dataframe, variable)
    dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
    dataframe.loc[(dataframe[variable] > up_limit), variable] = up_limit
```

In [40]:
```
for col in outliers_list:
    replace_with_thresholds(df, col)
```

In [41]:
```
outliers_boxplot(df, high_cols)
```



In [42]:
```
outliers_boxplot(df, low_cols)
```



Görüldüğü üzere değişkenler üzerindeki outlier gözlemler baskılandı.

2.1.2-Eksik Değerlerin Düzeltilmesi

In [43]:
```
missing_values_table(df, na_name=True)
```

```
        n_miss  ratio
SALARY      59 18.320
```

Out[43]:
```
['SALARY']
```

In [44]:
```
df.shape
```

Out[44]:
```
(322, 20)
```

In [45]:
```
df.dropna(inplace=True) # Missing value satırları silindi. SONRADAN DEĞİŞTİR !!!
```

In [46]:
```
df.shape  # missing value'ların atılması sonrası veri seti boyutu
```

Out[46]:
```
(263, 20)
```

2.2-Yeni Değişkenler Türetme (Feature Extraction)

Bu bölüm, Keşifçi Veri Analizi bölümünde ele alınan değişkenlerin bağımlı değişken ile olan ilişkisi üzerinden, yeni değişkenler türetme aşamasıdır.

Var olan değişkenler üzerinden türetilecek olan yeni değişkenlerle, bağımlı değişken ile arasında anlamlı çıkarımlar kurulması amaçlanacaktır.

In [47]:
```
df.head()
```

Out[47]:

	ATBAT	HITS	HMRUN	RUNS	...	ASSISTS	ERRORS	SALARY	NEWLEAGUE
1	315	81	7	24.000	...	43.000	10	475.000	N
2	479	130	18	66.000	...	82.000	14	480.000	A
3	496	141	20	65.000	...	11.000	3	500.000	N
4	321	87	10	39.000	...	40.000	4	91.500	N
5	594	169	4	74.000	...	404.500	23	750.000	A

5 rows × 20 columns

Bu bölümde türeteceğimiz yeni değişkenler için bağımlı değişkenle arasındaki yüksek korelasyon kriteri dikkate alınarak yapılacaktır.

2.2.1-Kategorik Değişkenler Türetme

Bağımsız değişkenlerden kategorik türde yeni değişkenlerin türetilmesi aşamasıdır. Değişkenler segmentlere ayrılacaktır.

YEARS Değişkeni ile Kategorik Değişken Türetme
Years: Oyuncunun major liginde oynama süresini ifade etmektedir. Oyuncunun, SALARY bağımlı değişkenine doğrudan ve diğer değişkenler üzerinde de dolaylı olarak etkileyebilecek bir değişkendir. Bir oyuncunun ilgili ligdeki skorlarını, tüm kariyerindeki skorları üzerinden yorumlanırken YEARS değişkeni göz önünde tutulacaktır.

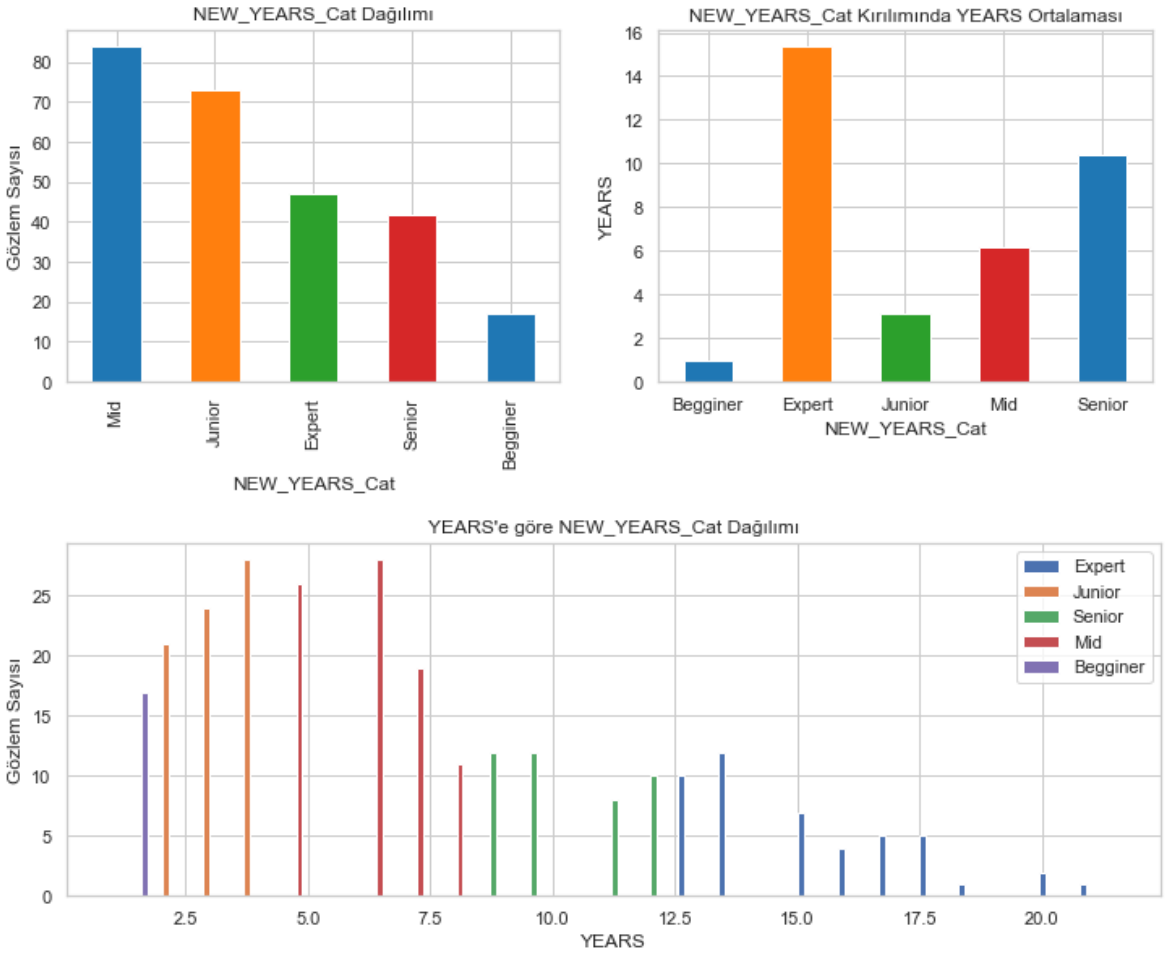
In [48]:
```
df["YEARS"].describe()
```

Out[48]:
```
count   263.000
mean      7.302
std       4.763
min       1.000
25%       4.000
50%       6.000
75%      10.000
max      21.500
Name: YEARS, dtype: float64
```

In [49]:
```
df.loc[(df["YEARS"] <= 1), "NEW_YEARS_Cat"] = "Begginer"
```

```
df.loc[(df["YEARS"] > 1) & (df['YEARS'] <= 4), "NEW_YEARS_Cat"] = "Junior"
df.loc[(df["YEARS"] > 4) & (df['YEARS'] <= 8), "NEW_YEARS_Cat"] = "Mid"
df.loc[(df["YEARS"] > 8) & (df['YEARS'] <= 12), "NEW_YEARS_Cat"] = "Senior"
df.loc[(df["YEARS"] > 12), "NEW_YEARS_Cat"] = "Expert"
```

In [50]:
```
cat_analyser_plot(df, "NEW_YEARS_Cat", "YEARS")
```



Grafikten anlaşılacağı üzere, veri setinde yeni deneyim sahibi(Begginer, Junior) oyuncular çoğunlukta. Buna karşın, beklenildiği gibi deneyimli oyuncuların SALARY ortalaması daha yüksek. Buradan, bağımlı değişkenin belirlenmesinde oyuncu deneyiminin etkili olduğu söylenebilir.

HITS Değişkeni ile Kategorik Değişken Türetme

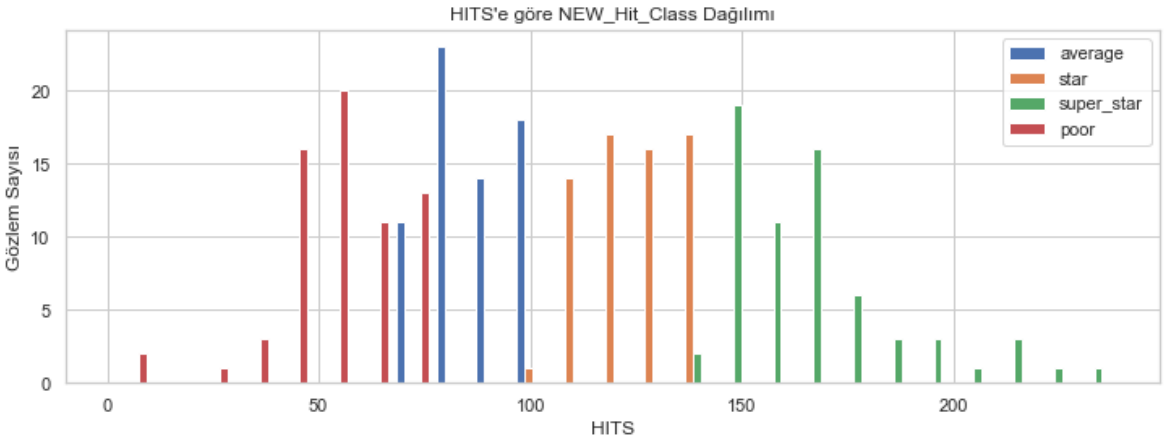
HITS: Oyuncunun 1986-1987 sezonundaki isabet sayısını ifade etmektedir. Bağımlı değişken(SALARY) üzerinde doğrudan etkili olan bu değişken farklı segmentlere ayrılarak bağımlı değişken üzerindeki anlamlılığı ayrıntılı bir şekilde gözlemlenecektir.

In [51]:
```
df["HITS"].describe()
```

Out[51]:
```
count   263.000
mean    107.829
std      45.125
min       1.000
25%      71.500
50%     103.000
75%     141.500
max     238.000
Name: HITS, dtype: float64
```

In [52]:
```
df["NEW_Hit_Class"] = pd.qcut(df['HITS'], 4, labels=['poor','average','star','super_star'])
```

In [53]:
```
cat_distribution_plot(df, "NEW_Hit_Class", "HITS")
```



Yeni türetilen NEW_Hit_class değişkenin dağılımını gösteren grafik. Dağılımın normal dağılıma yakın olduğu gözlenebilir. Buna marjinal kategorideki(poor, super_star) oyuncular azınlıkta.

LEAGUE Değişkeni ile Kategorik Değişken Türetme

LEAGUE: Oyuncunun sezon sonuna kadar oynadığı ligi gösteren A ve N seviyelerine sahip bir faktördür.

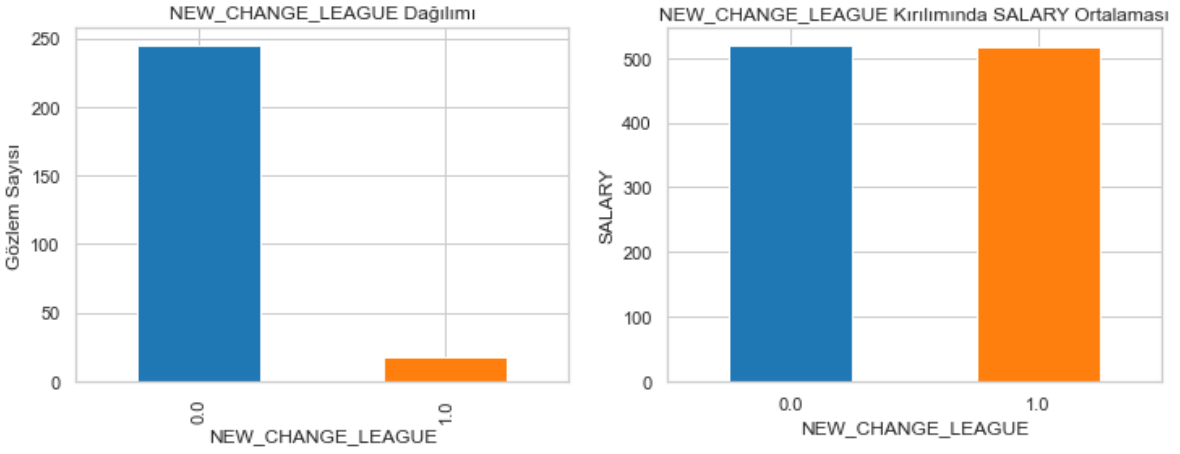
1986-1987 sezonu sonunda lig değiştiren oyuncular:

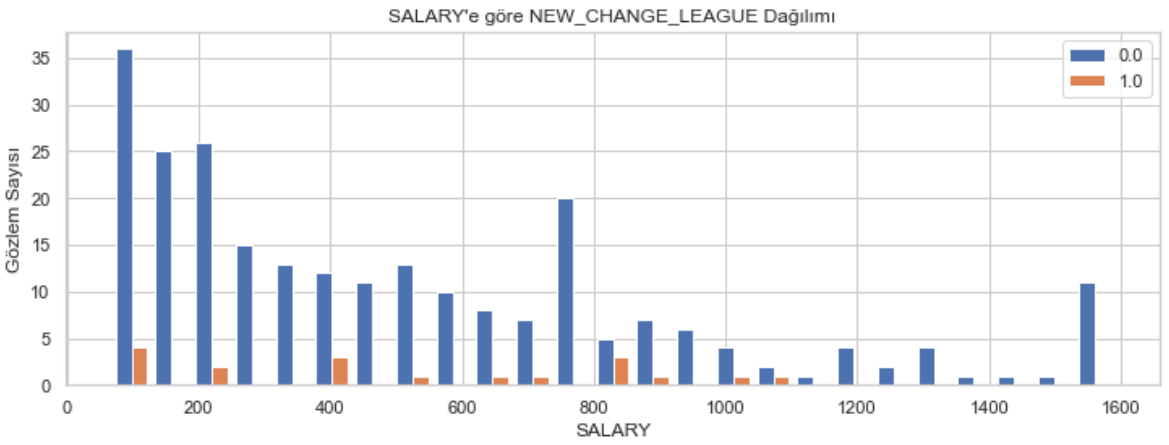
In [54]:
```
df.loc[(df["LEAGUE"] != df["NEWLEAGUE"]), "NEW_CHANGE_LEAGUE"] = 1
df.loc[(df["LEAGUE"] == df["NEWLEAGUE"]), "NEW_CHANGE_LEAGUE"] = 0
```

In [55]:
```
cat_summary(df, "NEW_CHANGE_LEAGUE", "SALARY")
```

```
       NEW_CHANGE_LEAGUE  Ratio  SALARY_Mean
0.000                245 93.156      520.170
1.000                 18  6.844      518.056
```

In [56]:
```
cat_analyser_plot(df, "NEW_CHANGE_LEAGUE", "SALARY")
```





Grafiklerden anlaşılacağı üzere, veri setinde sezon bitiminde lig değiştirmeyen oyuncular baskın bir şekilde çoğunluktayken, lig değiştiren azınlık oyuncularla ortalama aynı SALARY aldıklarını söyleyebiliriz. Buna göre, lig değişimi

bağımlı değişken için önemli bir faktör olabilir. Lig değiştiren oyuncu yüzdesi oldukça az olduğundan(%6) rare olarak anlamsız bir değişken olarak da görülebilir.

2.2.2-Nümerik Değişkenler Türetme

Bağımsız değişkenlerden nümerik türde yeni değişkenlerin türetilmesi aşamasıdır. Değişkenler arasındaki oranlar dikkate alınacaktır.

create_ratio_cols(), iki değişkenin oranını alıp isimlendirerek yeni değişken oluşturan bir fonksiyondur.

In [57]:
```
def create_ratio_cols(dataframe, numerator_col, denominator_col, new_col_name=False):
    if new_col_name:  # yeni dataframe'in adlandırılması fonksiyonda
        dataframe[new_col_name] = dataframe[numerator_col]/(dataframe[denominator_col]+0.001)
    else:             # Bölme sonucu paydanın sıfır olması durumunda sonsuz çıkmaması için 0.001
        dataframe[f"NEW_{numerator_col}/{denominator_col}"] = dataframe[numerator_col]/(dataframe
```

YEARS Değişkeni ile Nümerik Değişken Türetme

Oyuncunun kariyeri boyunca ortalama isabetli atış sayısı:

In [58]:
```
create_ratio_cols(df, "CHITS", "YEARS", "NEW_CRBI/YEARS")
```

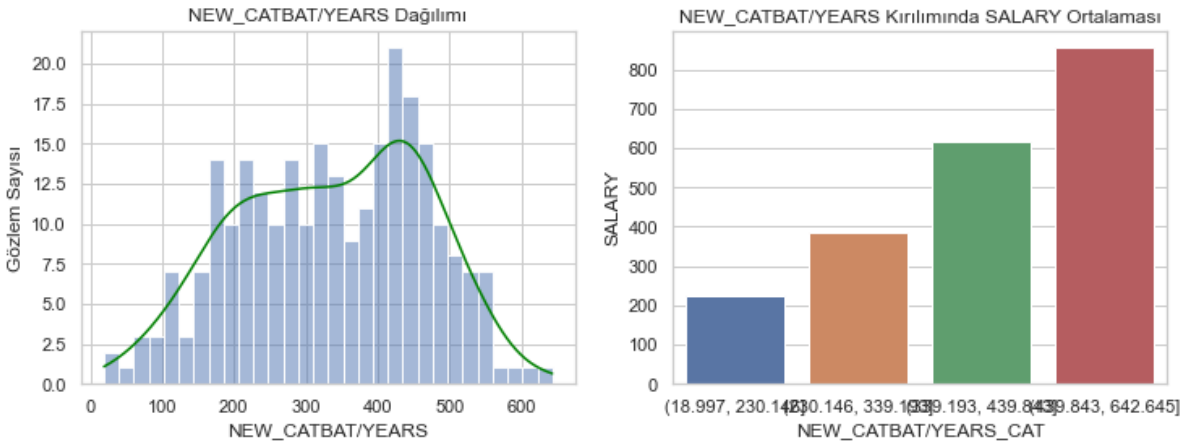
Oyuncunun kariyeri boyunca ortalama yaptığı en değerli atış sayısı:

In [59]:
```
create_ratio_cols(df, "CHMRUN", "YEARS")
```

Oyuncular ait diğer değişkenlerin zamana bağlı oranları nelerdir?

In [60]:
```
create_ratio_cols(df, "CHITS", "YEARS")
create_ratio_cols(df, "CRUNS", "YEARS")
create_ratio_cols(df, "CRBI", "YEARS")
create_ratio_cols(df, "CATBAT", "YEARS")
```

In [61]:
```
num_analyser_plot(df, "NEW_CATBAT/YEARS", "SALARY")
```



Yeni oluşturulan NEW_CATBAT/YEARS değişkenine ait grafikler. Buna göre, oyuncunun kariyeri boyunca topa vurma ortalamasının SALARY üzerinde doğrudan etkili olduğunu söyleyebiliriz.

HITS Değişkeni ile Nümerik Değişken Türetme

HITS: Oyuncunun 1986-1987 sezonundaki toplam isabet sayısını ifade etmektedir.

Oyuncunun, toplam topa vuruş sayısına göre isabet oranı nedir?

In [62]:
```
create_ratio_cols(df, "HITS", "ATBAT")
```

Oyuncunun kariyeri boyunca topa vurma sayısına göre isabetli vuruş oranı nedir?

In [63]:
```
create_ratio_cols(df, "CHITS", "CATBAT")
```

RUNS Değişkeninden Yeni Değişkenler Türetme

RUNS: Oyuncunun, 1986-1987 sezonunda takımına kazandırdığı sayı ifade etmektedir. Beklenildiği gibi, bağımlı

değişkenle olan korelasyonuna bakıldığında oyuncunun SALARY değerini etkileyen önemli bir parametre olduğu görülebilir. Bu değişkenle ilişkili başka değişkenler türetmek, kurulacak model için anlamlı bilgi taşıyabilir.

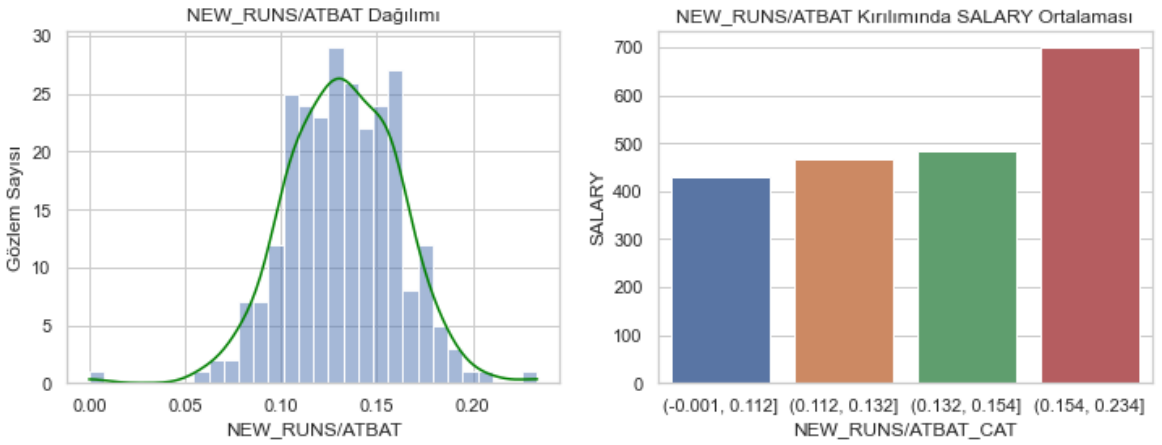
Oyuncunun 1986-1987 sezonunda topa yaptığı vuruş sayısına göre takımına kazandırdığı sayı oranı nedir?

In [64]:
```
create_ratio_cols(df, "RUNS", "CRUNS")
```

Oyuncunun, toplam topa yaptığı vuruş sayısına oranla takımına kazandırdığı sayı oranı nedir?

In [65]:
```
create_ratio_cols(df, "RUNS", "ATBAT")
```

In [66]:
```
num_analyser_plot(df, "NEW_RUNS/ATBAT", "SALARY")
```



Türetilen NEW_RUNS/ATBAT değişkeninin yukarıdaki grafikten dağılımı incelendiğinde, oyuncunun toplam topa vuruş sayısına göre takımına kazandırdığı sayı oranı arttıkça ortalama maaşı(SALARY) artmaktadır.

Oyuncunun kariyeri boyunca yaptığı isabetli atışlardan kaçı takımına sayı kazandırdı?

In [67]:
```
create_ratio_cols(df, "CRUNS", "CHITS")
```

Oyuncunun ilgili sezonda takımına kazandırdığı sayılara oranla, en değerli vuruş sayısı nedir?

In [68]:
```
create_ratio_cols(df, "HMRUN", "RUNS")
```

Oyuncunun kariyeri boyunca takımına kazandırdığı sayılara oranla, kariyeri boyunca yaptığı en değerli vuruş sayısı nedir?

In [69]:
```
create_ratio_cols(df, "CHMRUN", "CRUNS")
```

RBI Değişkeninden Yeni Değişkenler Türetme

RBI: Vurucu rolündeki oyuncunun vuruş yaptıgında koşu yaptırdığı oyuncu sayısını ifade etmektedir. Değişkenler arasındaki korelasyondan anlaşılacağı üzere bir oyuncunun koşu yaptırdığı oyuncu sayısı, SALARY değişkeninde önemli bir faktör. Buna göre, RBI değişkeni kullanılark yeni değişkenler türetilerek anlamlılığı irdelenecek.

Oyuncunun kariyerinde koşu yaptırdığı oyuncu sayısına göre sezondaki sayısı nedir?

In [70]:
```
create_ratio_cols(df, "RBI", "CRBI")
```

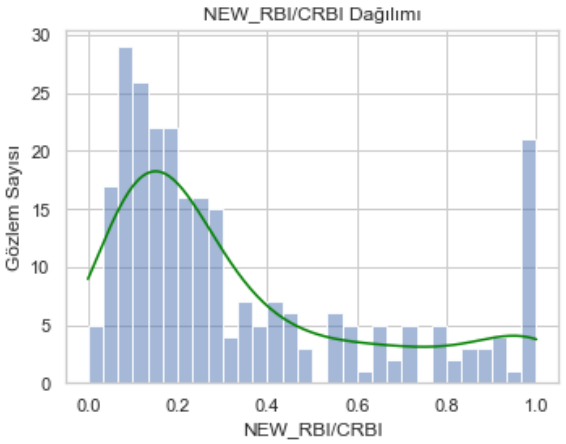
Oyuncunun, kariyerindeki isabetli atışlarından koşu yaptırdığı oyuncu oranı nedir?

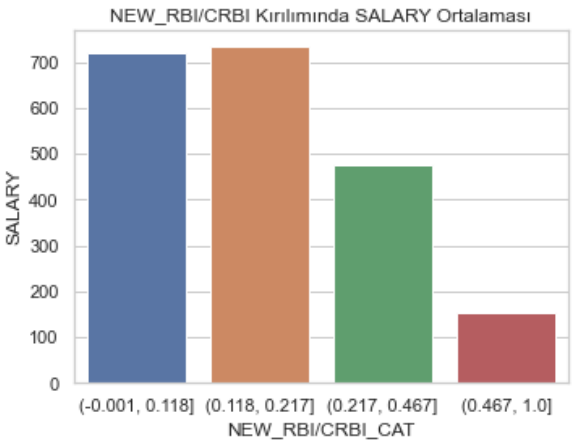
In [71]:
```
create_ratio_cols(df, "CRBI", "CHITS") # Sorun çıkmadı
```

Oyuncunun, 1986-1987 sezonunda topa yaptığı vuruş sayısına göre koşu yaptırdığı oyuncu sayısı nedir?

In [72]:
```
create_ratio_cols(df, "RBI", "ATBAT") # arttırdı ama kalabilir
```

In [73]:
```
num_analyser_plot(df, "NEW_RBI/CRBI", "SALARY")
```





Grafiklere göre, oyuncunun sezondaki toplam topa vuruş sayısına göre koşu yaptırdığı oyuncu oranı arttıkça SALARY değeri azalmaktadır. Türetilen buu değişkenle bağımlı değişken arasında negatif yönlü doğrusal bir ilişki görülmektedir.

WALKS Değişkeninden Yeni Değişkenler Türetme

WALKS: Karşı oyuncuya yaptırılan hata sayısını ifade etmektedir.

Oyuncunun, ilgili yıllarda kariyerine göre karşı oyuncuya yaptırdığı hata sayısı oranı. (İlgili sezonda karşı takıma yaptırdığı hata sayı / kariyerindeki toplam yaptırdığı hata sayısı)

In [74]:
```
create_ratio_cols(df, "WALKS", "CWALKS")
```

Oyuncunun, karşı oyuncuya yaptırdığı hata sayısına bağlı olarak en değerli vuruş sayısı oranı:

In [75]:
```
create_ratio_cols(df, "WALKS", "HMRUN")
```

Oyuncunun kariyeri boyunca karşı takım oyuncusuna yaptırdığı hata ortalaması nedir?

In [76]:
```
create_ratio_cols(df, "CWALKS", "YEARS")
```

ATBAT Değişkeninden Yeni Değişkenler Türetme

ATBAT: Oyuncunun 1986-1987 sezonunda bir beyzbol sopası ile topa yaptığı vuruş sayısını ifade etmektedir.

Oyuncunun kariyeri boyunca yaptığı atışa göre 1986-1987 sezonunda yaptığı atış oranı nedir?

In [77]:
```
create_ratio_cols(df, "ATBAT", "CATBAT")
```

Oyuncun 1986-1987 sezonunda yaptığı hataların kaçı vuruşlarından kaynaklıdır?

In [78]:
```
create_ratio_cols(df, "ATBAT", "ERRORS")
```

Diğer Değişkenlerden Yeni Değişkenler Türetme

Oyuncunun kariyerine göre, toplam en değerli vuruş sayısının ilgili sezondakine oranı nedir?

In [79]:
```
create_ratio_cols(df, "HMRUN", "CHMRUN")
```

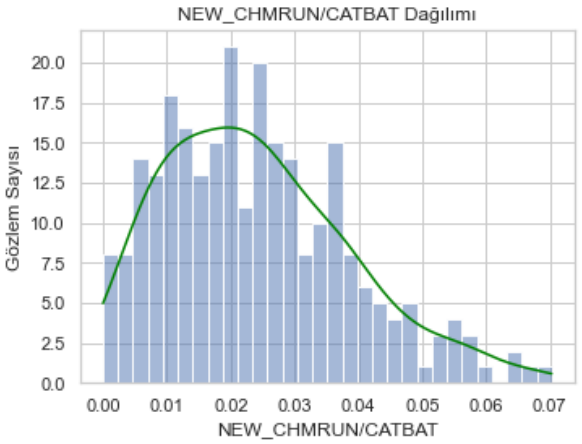
Oyuncunun kariyerine göre, toplam isabetli vuruş sayısının ilgili sezondakine oranı nedir?

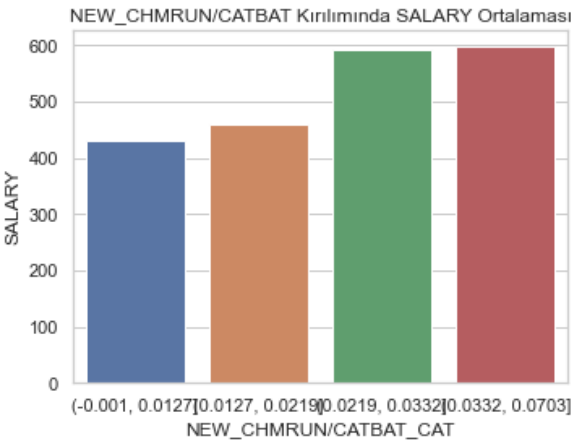
In [80]:
```
create_ratio_cols(df, "CHMRUN", "CHITS")
```

Oyyncunun kariyeri boyunca topa vurma sayısına oranla yaptığı en değerli vuruş sayısı kaçtır?

In [81]:
```
create_ratio_cols(df, "CHMRUN", "CATBAT")
```

In [82]:
```
num_analyser_plot(df, "NEW_CHMRUN/CATBAT", "SALARY")
```





Oyuncunun toplam topa vuruşlarına göre en değerli sayısı oranı doğrudan SALARY üzerinde etkisi baskın gözükmese de dağılımdaki çarpıklık anlamlılık taşıyor olabilir.

2.3-Encoding işlemlerini gerçekleştirme

Bu bölüm, veri setindeki kategorik değişkenlerin model tarafından anlaşılabilir formatta olacak şekilde encode edilmesi aşamasıdır. Sırasıyla Label Encoding, Rare Encoding ve One-Hot Encoding işlemleri yapılacaktır.

2.3.1-Label Encoder

2 farklı deişkene sahip kategorik değişkenlerin binary(0-1) haline çevrilmesi aşamasıdır.

In [83]:
```
def label_encoder(dataframe, binary_col):
    labelencoder = LabelEncoder()
    dataframe[binary_col] = labelencoder.fit_transform(dataframe[binary_col])
    return dataframe
```

In [84]:
```
binary_cols = [col for col in df.columns if df[col].dtype == "O" and df[col].nunique() == 2] #
binary_cols
```

Out[84]:
```
['LEAGUE', 'DIVISION', 'NEWLEAGUE']
```

In [85]:
```
for col in binary_cols:
    df = label_encoder(df, col)
df.head(2)
```

Out[85]:

	ATBAT	HITS	HMRUN	RUNS	...	NEW_ATBAT/ERRORS	NEW_HMRUN/CHMRUN	NEW_CHMRUN/CHITS	NEW_CHMRUN/C
1	315	81	7	24.000	...	31.500	0.101	0.083	
2	479	130	18	66.000	...	34.214	0.286	0.138	

2 rows × 46 columns

2.3.2-Rare Encoding

Bağımlı değişkene göre dağılımı seyrek olan değişken gözlemlerinin birarada encode edilmesidir.

In [86]:
```
cat_cols, num_cols, cat_but_car = grab_col_names(df)
```

In [87]:
```
def rare_analyser(dataframe, target, cat_cols):
    for col in cat_cols:
        print(col, ":", len(dataframe[col].value_counts()))
        print(pd.DataFrame({"COUNT": dataframe[col].value_counts(),
                            "RATIO": dataframe[col].value_counts() / len(dataframe),
                            "TARGET_MEAN": dataframe.groupby(col)[target].mean()}), end="\n\n\n")
```

In [88]:
```
rare_analyser(df, "SALARY", cat_cols)
```

```
NEW_YEARS_Cat : 5
           COUNT  RATIO  TARGET_MEAN
Begginer      17  0.065      149.647
```

```
Expert       47  0.179      741.028
Junior       73  0.278      236.212
Mid          84  0.319      625.143
Senior       42  0.160      705.686


LEAGUE : 2
   COUNT  RATIO  TARGET_MEAN
0    139  0.529      524.453
1    124  0.471      515.062


DIVISION : 2
   COUNT  RATIO  TARGET_MEAN
0    129  0.490      594.257
1    134  0.510      448.563


NEWLEAGUE : 2
   COUNT  RATIO  TARGET_MEAN
0    141  0.536      519.815
1    122  0.464      520.268


NEW_Hit_Class : 4
            COUNT  RATIO  TARGET_MEAN
poor           66  0.251      313.232
average        66  0.251      367.457
star           65  0.247      591.736
super_star     66  0.251      808.762


NEW_CHANGE_LEAGUE : 2
       COUNT  RATIO  TARGET_MEAN
0.000    245  0.932      520.170
1.000     18  0.068      518.056
```

Rare olarak kabul edilebilecek (ratio < 0.01 seviyede) herhangi bir değişken gözlemi gözükmüyor.

2.3.3-One-Hot Encoding

2'den fazla benzersiz(uniq) gözleme sahip kategorik değişkenlerin encode edilmesi aşamasıdır.

In [89]:
```
ohe_cols = [col for col in df.columns if 10 >= df[col].nunique() > 2]  # One-Hot encoding için uy
ohe_cols
```

Out[89]:
```
['NEW_YEARS_Cat', 'NEW_Hit_Class']
```

In [90]:
```
def one_hot_encoder(dataframe, categorical_cols, drop_first=True):
    dataframe = pd.get_dummies(dataframe, columns=categorical_cols, drop_first=drop_first)
    return dataframe
```

In [91]:
```
df = one_hot_encoder(df, ohe_cols)
```

2.3.4-Numerik Değişkenler için Standartlaştırma İşlemi

Lineer Regresyon uzaklık temelli bir model tekniği olduğundan, değişkenler arasındaki yüksek oranların yanlılık yaratmaması için belirli seviyede standartlaştırılması gerekmektedir. Bu aşamada nümerik değişkenlerin standartlaştırılması yapılacaktır.

In [92]:
```
cat_cols, num_cols, cat_but_car = grab_col_names(df)  # Nümerik değişkenleri kullanacağız.
```

In [93]:
```
num_cols.remove("SALARY")
```

In [94]:
```
scaler = StandardScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])
```

In [95]:
```
df.head(3)    # Encoding sonrası veri seti
```

Out[95]:

	ATBAT	HITS	HMRUN	RUNS	...	NEW_YEARS_Cat_Senior	NEW_Hit_Class_average	NEW_Hit_Class_star	NEW_Hit_Cla

1	-0.603	-0.596	-0.530	-1.207	...	0	1	0
2	0.513	0.492	0.738	0.442	...	0	0	1
3	0.628	0.736	0.969	0.403	...	1	0	1

3 rows × 51 columns

In [96]:
```
df.shape
```

Out[96]: (263, 51)

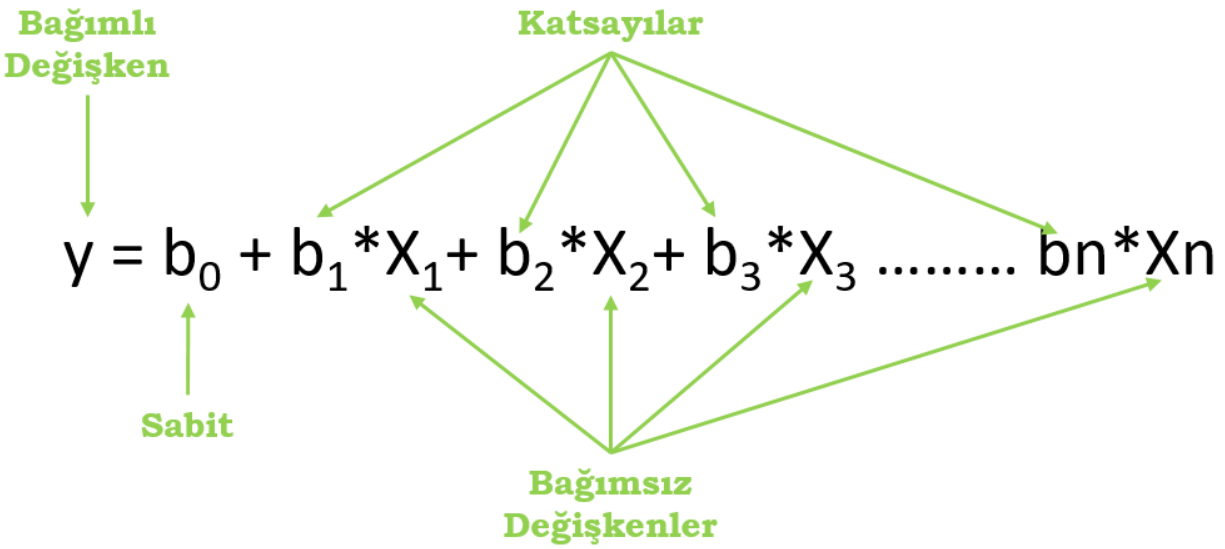
Modelden oluşturmadan önce, Feature Extraction aşamasında dataframe değişken sayısının 20'den 49'a çıkarıldığı görülebilir.

3-Model oluşturma ve Test

Bu aşamada, Keşifçi Veri Analizi ve Özellik Mühendisliği bölümlerinde hazırlanan veri seti üzerinde Makine Öğrenmesinin temeli olarak sayılan Lineer Regresyon tekniğini kullanarak model geliştireceğiz. Giriş düzeyde tahminleme modeli geliştirilecek olup, konunun detaylarına(model metrik optimizasyonu) girilmeden, test süreci gerçekleştirilecektir. Sonuç olarak, hazırlanan veri seti ile model metriklerinin sonuçları ve model için anlamlı/önemli değişkenler incelenecektir.

3.1-Lineer Regresyon Modeli Kurma

lineer Regresyon, bağımlı ve bağımsız değişken arasındaki ilişkiye dayanarak çıkarılan birinci dereceden matematiksel denklemi(model) ifade eder.



Fotoğraf: www.veribilimiokulu.com - Çoklu Regresyon

Yukarıdaki denklemden görüleceği üzere, çok değişkenli bir regresyon modeli matematiksel olarak bu şekilde ifade edilir. Her bir bağımsız değişkene çarpım durumundaki katsayılar(coefficients) ise değişkenin bağımlı değişkenle olan ilişkisinin ağırlığını ifade etmektedir. b0 olarak ifade edilen sabit(bias/intercept) sayısı ise denklemin en uygun şekliyle kurulmasına katkı sağlamaktadır.

In [97]:
```
y = df["SALARY"]  # Bağımlı değişken
X = df.drop(["SALARY"], axis=1)  # Bağımsız değişkenler
```

In [98]:
```
#cat_cols, num_cols, cat_but_car = grab_col_names(df)  # Nümerik değişkenleri kullanacağız.
#num_cols.remove("SALARY")
#scaler = StandardScaler()
#df[num_cols] = scaler.fit_transform(df[num_cols])
```

In [99]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=17)  # Ver
```

In [100…]:
```

```

```
reg_model = LinearRegression().fit(X_train, y_train)  # Model kuruldu.
```

In [101…

```
print('intercept:', reg_model.intercept_) # b0 sabit sayısı
```

```
intercept: 417.0811991295161
```

In [102…

```
print('coefficients:', reg_model.coef_)    # Denklem katsayıları
```

```
coefficients: [-204.00179284   16.76121988   51.70880592  151.36725278   60.29818804
  -38.84038793 -421.47145652 -339.95881784  345.15873402   36.48347872
  230.25261531   60.9579353    60.1947167   -16.24534323  -48.49017197
   64.15777208   26.19627364   -6.65732131   34.07217167  105.600061
 -129.35886296   46.08855603  433.75254302   29.18628952 -508.77226324
  -61.66926865  -43.97645214 -188.60537035  -88.4240921   -81.35076079
  -35.89069404 -223.5976539  -165.46961373   61.91987495  -13.28048812
  105.88849242   65.61666324   58.4103033   -92.08843707    4.10816672
  115.8815059  -104.35386856  312.48655211  125.66428251 -291.4235547
  -41.08472311   70.41659623   82.70673845  276.67968105  374.86585672]
```

Model Denklemi

In [103…

```
print("Model Denklemi:\t"+
f"Y= {reg_model.intercept_:.2f}{reg_model.coef_[0]:.2f}X{reg_model.coef_[0]:.2f}X+------+{reg_mode
```

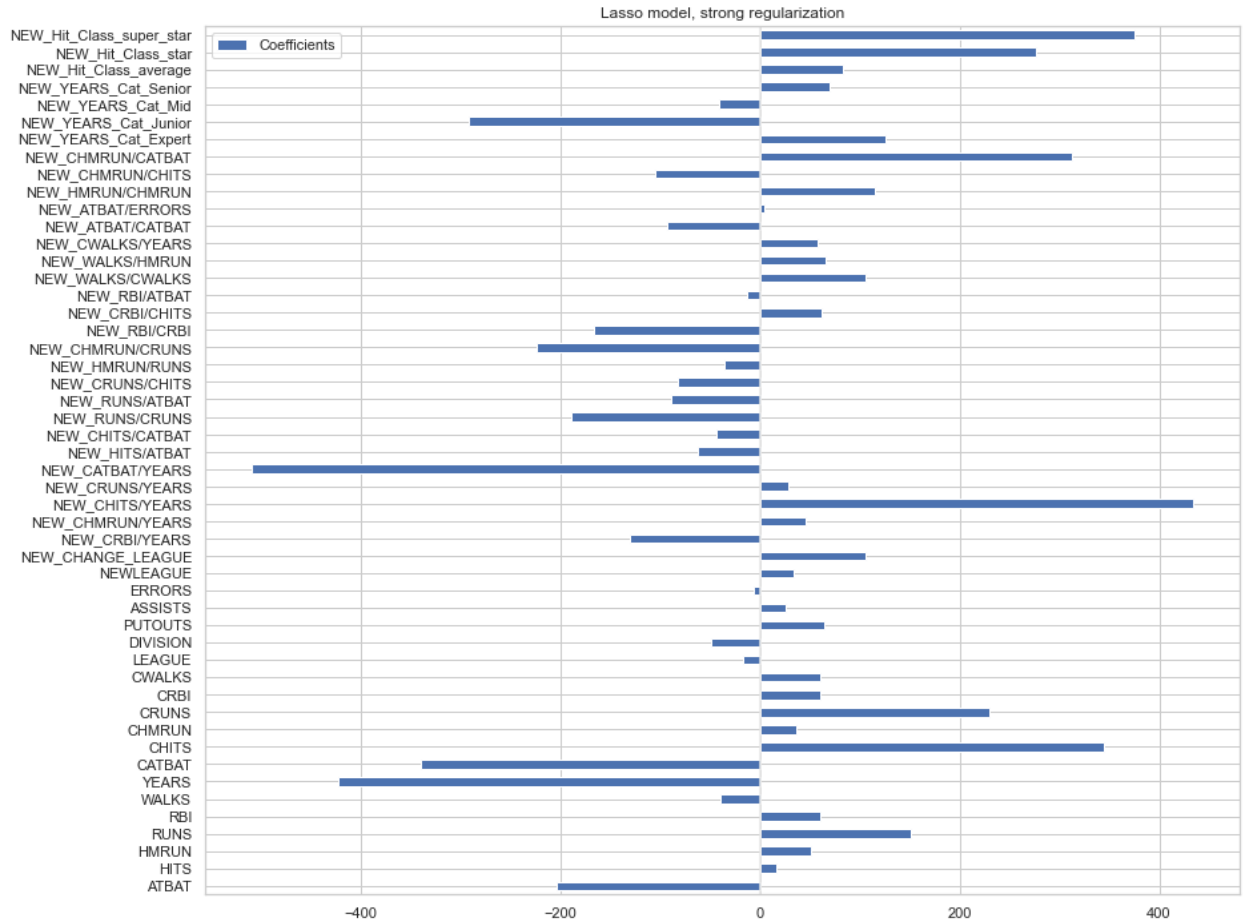
```
Model Denklemi: Y= 417.08-204.00X-204.00X+------+374.87X
```

Model denklemi yukardaki gibi elde edildi. Buna göre, bağımsız değişken değerlerini öğrendiğimiz bir örnek senaryoda oyuncunun SALARY değerine ulaşabiliriz.

Katsayıları, ağırlıklarına göre grafiksel olarak gösterecek olursak:

In [104…

```
coefs = pd.DataFrame( reg_model.coef_, columns=['Coefficients'], index=X_train.columns)
coefs.plot(kind='barh', figsize=(18, 12))
plt.title('Lasso model, strong regularization')
plt.axvline(x=0, color='.9')
plt.subplots_adjust(left=.3)
```



Buna göre, yeni türetilen değişkenlerin ağırlık değerlerinden, modelin sağlıklı kurulmasında önemli ölçüde pay aldığı görülebilir.

Elde edilen katsayılardan, bağımlı değişkenle pozitif veya negatif yönlü ağırlığı en yüksek 5'er katsayıyı görmek istersek:

Pozitif yönlü yüksek ağırlıklı ilk 5 değişken:

In [105…
```
coefs.sort_values("Coefficients", ascending=False).head(5)
```

Out[105…

	Coefficients
NEW_CHITS/YEARS	433.753
NEW_Hit_Class_super_star	374.866
CHITS	345.159
NEW_CHMRUN/CATBAT	312.487
NEW_Hit_Class_star	276.680

Negatif yönlü yüksek ağırlıklı ilk 5 değişken:

In [106…
```
coefs.sort_values("Coefficients").head(5)
```

Out[106…

	Coefficients
NEW_CATBAT/YEARS	-508.772
YEARS	-421.471
CATBAT	-339.959
NEW_YEARS_Cat_Junior	-291.424
NEW_CHMRUN/CRUNS	-223.598

Her iki sıralamada da veri setininde var olan değişkenlerin yanı sıra, daha sonra türetmiş olduğumuz değişkenlerin çoğunlukta olduğunu görmek mümkün.

3.2-Tahminleme Testi

Bu aşamada, veri setinden rastgele bir oyuncu seçilecek olup bu oyuncunun SALARY bilgisi oluşturulan Lineer Regresyon modeliyle tahmin edilmeye çalışılacaktır.

Birinci rastgele oyuncu

In [107…
```
random_user = X.sample(1, random_state=7)  # Veri setinden rastgele bir oyuncu seçer
```

In [108…
```
reg_model.predict(random_user)   # Oyuncunun bilgilerine göre tahmin edilen SALARY değeri
```

Out[108… `array([524.28805224])`

In [109…
```
view_df = load_hitters()   # Oyuncu bilgilerini göstermek için
view_df.loc[random_user.index]  # Random seçilen oyuncuya dair bilgiler.
```

Out[109…

	AtBat	Hits	HmRun	Runs	...	Assists	Errors	Salary	NewLeague
118	408	94	4	42	...	487	19	535.000	N

1 rows × 20 columns

İkinci rastgele oyuncu

In [110…
```
random_user = X.sample(1, random_state=8)  # Veri setinden rastgele bir oyuncu seçer
```

In [111…
```
reg_model.predict(random_user)  # Oyuncunun bilgilerine göre tahmin edilen SALARY değeri
```

Out[111… `array([255.25215795])`

In [112…
```
view_df = load_hitters()
view_df.loc[random_user.index]
```

Out[112…

	AtBat	Hits	HmRun	Runs	...	Assists	Errors	Salary	NewLeague

146	160	39	8	18	...	3	0	275.000		A

1 rows × 20 columns

Görüldüğü üzere tahminleme belirli oranda doğru çalışıyor. "Belirli oran" nedir?

3.3-Tahmin Başarısını Değerlendirme

Bu başlık altında geliştirmiş olduğumu lineer regresyon modelinin başarısını değerlendireceğiz. Bunun için R-Kare ve Root Mean Square Error(RMSE) testini kullanacağız.

Train Veri Setinde R-Kare Testi:

Bu test, geliştirilen lineer regresyon modelinde bağımsız değişkenlerin bağımlı değişkeni açıklama yüzdesini ifade eder. Regresyon problemlerinde modelin başarısını/performansını değerlendirmek için kullanılan bir ölçüttür. Bu ölçütü, train ve test veri setleri üzerinde ayrı ayrı olarak test edeceğiz.

$$\text{Adj. } R^2 = 1 - (1 - R^2)\frac{N-1}{N-k-1}$$

Düzeltilmiş R-Kare Formülü - www.istmer.com

In [113…
```
reg_model.score(X_train, y_train) # Train veri seti üzerinde R-Kare sonucu
```

Out[113… 0.7928712213603755

Train Veri Setinde RMSE Testi:

Root Mean Square Error(RMSE) testi, bir modelin tahmin ettiği değerlerle gerçek değerleri arasındaki uzaklık farkına bağlı olarak hatanın büyüklüğünü ölçen bir metriktir.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

Root Mean Square Error formülü - www.geeksforgeeks.org

In [114…
```
y_pred = reg_model.predict(X_train)     # Train veri seti üzerinde RMSE sonucu
np.sqrt(mean_squared_error(y_train, y_pred))
```

Out[114… 181.97151139109602

Test Veri Setinde R-Kare Testi:

In [115…
```
reg_model.score(X_test, y_test)  # Test veri seti üzerinde R-Kare sonucu
```

Out[115… 0.7261747926352417

Train Veri Setinde RMSE Testi:

In [116…
```
y_pred = reg_model.predict(X_test)     # Test veri seti üzerinde RMSE sonucu
np.sqrt(mean_squared_error(y_test, y_pred))
```

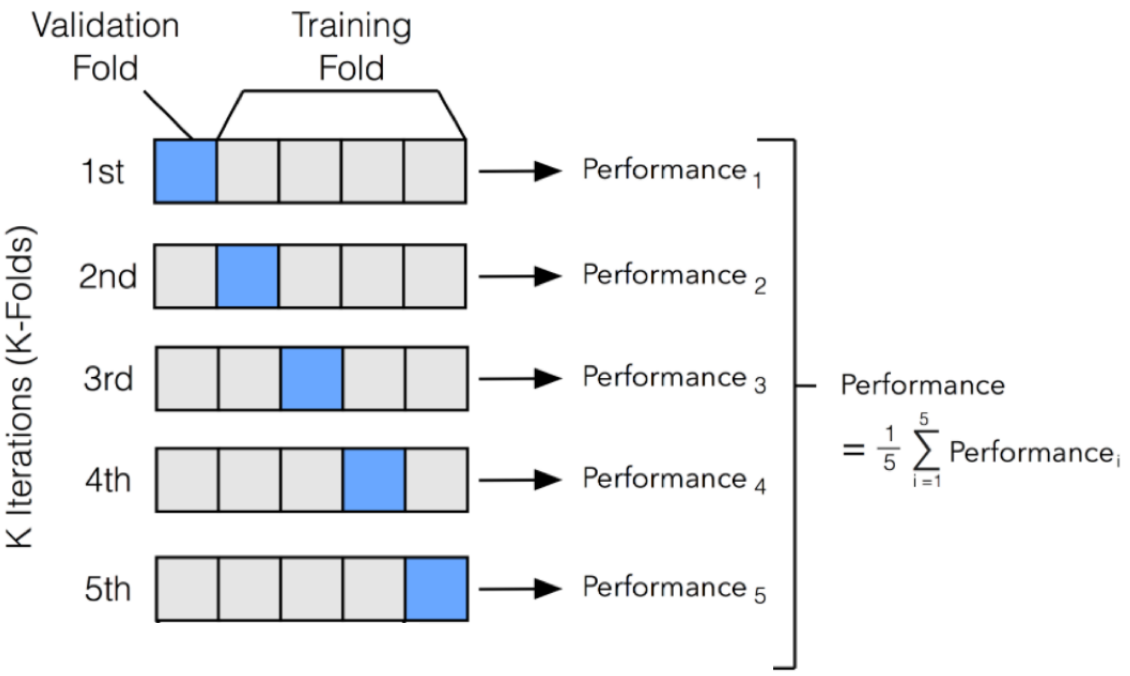
Out[116… 209.903572306384

Görüldüğü üzere, her iki ölçüm metiriği de train ve test veri setlerinde farklı sonuçlar gösterdi. Bu durumun sebebi, modeli oluştururken rastgele bir şekilde veri setinin ayrılması. farklı random_state değerleri için model başarı metrikleri sürekli farklı sonuçlar verecektir. Bu durumun önüne geçilmesi için Cross Validation yöntemiyle RMSE değerine bakılacaktır.

Cross Validation Tekniğiyle RMSE Testi

Cross-validation, makine öğrenmesi modelinin görmediği veriler üzerindeki performansını mümkün olduğunca objektif ve doğru bir şekilde değerlendirmek için kullanılan istatistiksel bir yeniden örnekleme(resampling) yöntemidir. (Cross-Validation nedir? Nasıl çalışır? - Merve Bayram Durna)

Veri seti küçük boyutlu olduğundan Cross Validation ile model sonucunu değerlendirmek daha objektif sonuç verecektir.



Fotoğraf: zitaoshen.rbind.io - 3 min of Machine Learning: Cross Vaildation

In [117…
```
np.mean(np.sqrt(-cross_val_score(reg_model, X, y, cv=10, scoring="neg_mean_squared_error")))  # 1
```

Out[117…
```
243.11784240045804
```

4-Sonuç

Elde edilen sonuçlar incelendiğinde, veri setindeki değişkenler ve yeni türetilen değişkenler ile oluşturulan lineer regresyon modeli, ortalama %75 oranında başarı ve 243 RMSE hata ölçüm değeri sonucunu vermiştir. Buna göre, 1986-1987 sezonunda oynamış oyuncuların belirli özellikleri modele gösterilerek oyuncunun aldığı maaş bilgisini tahmin edebiliriz. Bu problem, günümüz spor liglerindeki verilerilerle güncellenerek oyuncu maaş tahminine dayalı iş modelleri geliştirlebilir.

Modelin iyileştirilmesi için Decision Tree veya Random Forest gibi gelişmiş makine öğrenmesi modelleri kullanılabilir.