Proje Başlığı

Futbolcu İstatistiklerine Dayalı Makine Öğrenmesi ile Piyasa Değer Tahmini

Proje Konusu

Futbol kulüplerinin ekonomisi göz önüne alındığında doğru oyuncu fiyatlandırması kulüpler ve yatırımcılar için kritik öneme sahiptir. Futbolcuların piyasa değeri belirlenirken genellikle subjektif değerlendirme yapılmaktadır. Ancak bunun veri odaklı yapılması gerekmektedir. Bu proje, futbolcuların performans metriklerine dayalı olarak piyasa değerlerini tahmin eden bir makine öğrenmesi modeli geliştirmeyi amaçlıyor.

Proje İçeriği

Veri Toplama

- **Veri Kaynakları**: Futbolcu performans istatistikleri (oyuncu rating değeri, dribbling, kısa pas, uzun pas, pozisyon alma yeteneği, oyun içi reaksiyon vb.), oyuncuların yaş, pozisyon, kulüp bilgileri.
- **Veri Seti Hazırlama**: Çeşitli liglerden futbolcu verilerinin derlenmesi ve eksik/veri temizleme işlemleri.

Veri Analizi ve Özellik Seçimi

- Performans metriklerinin analiz edilerek en önemli özelliklerin belirlenmesi.
- Futbolcu pozisyonlarına göre oyuncuların ayrılması
- Modeli kötü etkileyecek özelliklerin çıkarılması
- Metinsel verilerin sayısal verilere çevrilmesi

Model Geliştirme

- **Makine Öğrenmesi Algoritmaları**: Doğrusal Regresyon, Karar Ağacı, Rastgele Orman, K-En Yakın Komşu, Yapay Sinir Ağları, Destek Vektör Makineleri gibi farklı algoritmaların değerlendirilmesi.
- **Model Eğitimi**: Verilerin eğitim ve test setlerine bölünmesi, modelin eğitilmesi ve parametre optimizasyonu.

Model Değerlendirme

- Model performansının doğruluk, hata oranı (MSE, RMSE), ve R-kare gibi metriklerle değerlendirilmesi.
- Model sonuçlarının mevcut piyasa değerleriyle karşılaştırılarak analiz edilmesi.

Sonuçlar ve Tartışma

- Modelin başarı düzeyi, öngörülen piyasa değerlerinin gerçek değerlerle ne kadar örtüştüğü.
- Veri odaklı fiyatlandırma yaklaşımının kulüpler ve yatırımcılar için potansiyel faydaları.

Sonuç ve Öneriler

- Modelin güçlü ve zayıf yönlerinin değerlendirilmesi.
- Gelecekteki çalışmalar için öneriler: Daha geniş veri setleri, Eş zamanlı futbolcu değerlendirmesi

Kullanım Alanı

- Futbol kulüpleri ve menajerler, transfer stratejilerini optimize etmek için bu modeli kullanabilir.
- Spor analistleri ve finans uzmanları, piyasa trendlerini öngörmek için modelden faydalanabilir.

Ulusal veya Uluslararası Benzer Örnekler

- **Transfermarkt** gibi platformlar oyuncu piyasa değerlerini manuel olarak belirlerken, bu proje verilere dayalı objektif tahmin sunar
- **CIES** ise yine manuel olarak bir piyasa değer aralığı belirler.

Kullanılan Teknolojiler, Diller ve Platformlar

- **Programlama Dilleri**: Python
- **Kütüphaneler**: scikit-learn, pandas, matplotlib, seaborn, tensorflow, keras, kfold
- **Platformlar**: Jupyter Notebook, Visual Studio Code

Proje Çıktısının Kullanım Alanları

- **Spor Kulüpleri**: Transfer politikalarının iyileştirilmesi
- **Medya ve Spor Analizi**: Oyuncu performansına dayalı analiz ve haber içerikleri

Proje Sürecinde Yaşanan Zorluklar

- **Veri Eksikliği**: Bazı futbolcu eksiklikler bulunmaktaydı. Örneğin milli takıma gitmemiş oyuncuların milli takım değerleri boştu.
- **Model İyileştirme**: Farklı algoritmaların denenmesi ve en iyi performansı veren modelin seçilmesi.
- **Gerçek Zamanlı Veri Girişi Eksikliği**: Proje sürecinde, modeli dışarıdan girilen verilerle test etmek mümkün olmuş, ancak anında veri girişi ve işlem desteği henüz entegrasyonu sağlanamamıştır.

Öğrenilenler ve Kazanımlar

- **Veri Bilimi**: Büyük veri setleriyle çalışma ve veri temizleme.
- **Makine Öğrenmesi**: İleri seviye algoritmaları kullanma ve model performansını iyileştirme.

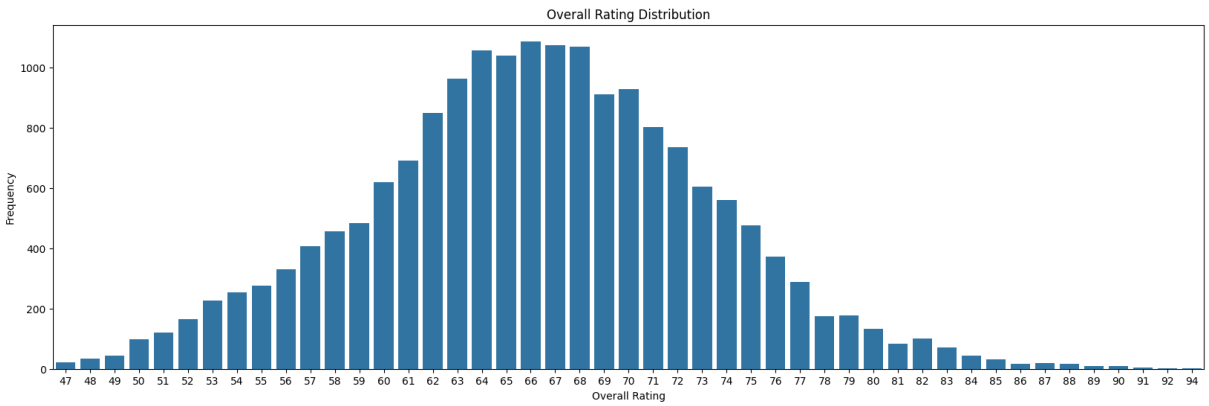
Proje İçin Faydalanılan Kaynaklar ve Web Siteleri

- www.kaggle.com Football Players Data
- Singh, H Understanding Random Forests: One Algorithm at a Time. Towards Data Science
- Tirendaz Akademi TensorFlow ve Keras ile Yapay Sinir Ağları, Medium
- Domino Data Lab ‘Sklearn’ Domino Data Science Dictionary

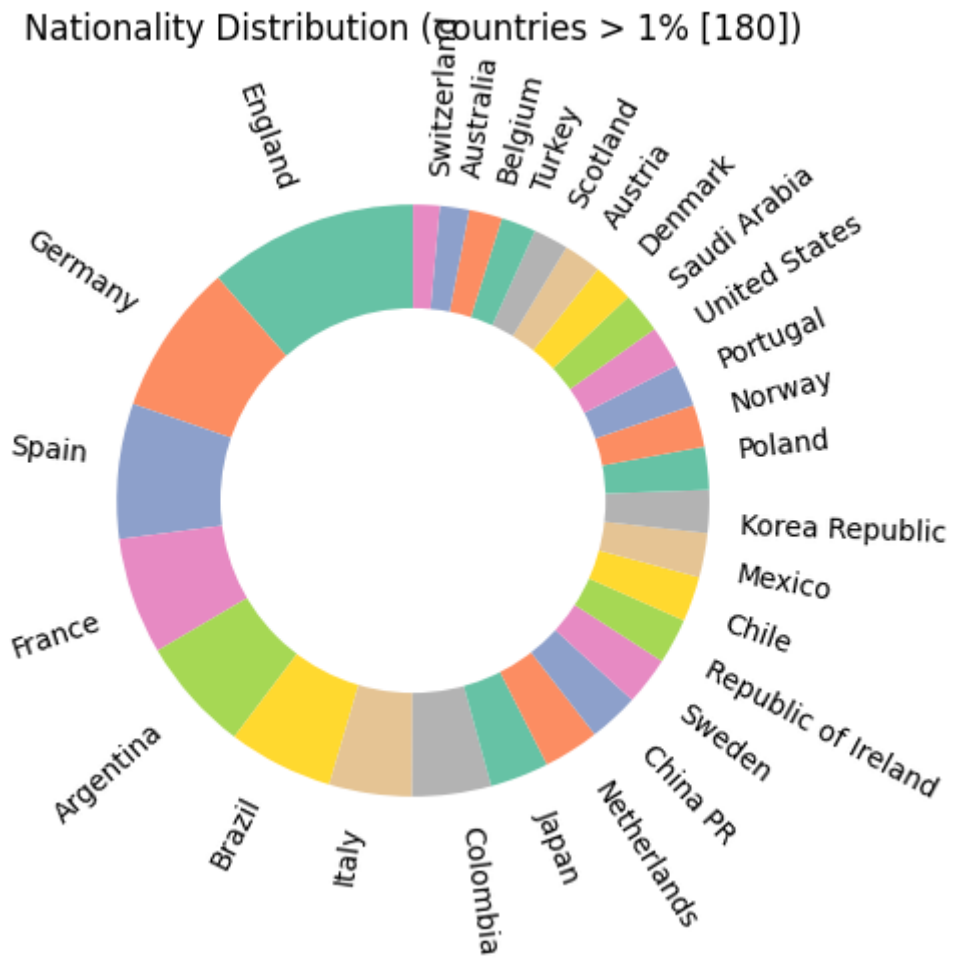
Veri Analizi ve Ön İşleme

```
from sklearn.preprocessing import LabelEncoder
df = df.drop(columns=['full_name','name','birth_date','nationality'])

categorical_to_nums = {'preferred_foot'  :   {'Left'  : 0, 'Right' : 1,},
                       'body_type'  :   {'Lean'  : 0, 'Normal' : 1, 'Stocky' : 2}}
for column, mapping in categorical_to_nums.items():
    temp_series = df[column].copy()
    null_mask = temp_series.isnull()
    temp_series.loc[~null_mask] = temp_series.loc[~null_mask].map(mapping)
    df[column] = temp_series.fillna(-1).astype('int64')

df['positions'] = df['positions'].str.split(',').str[0]


for column in ['height_cm','value_euro','wage_euro','release_clause_euro' ]:
    df[column] = df.groupby('positions')[column].transform(lambda x: x.fillna(x.median()))



df = df.drop(columns=['national_team','national_team_position','national_jersey_number','national_rating'])


encoder = LabelEncoder()

df['positions_encoded'] = encoder.fit_transform(df['positions'])


df = df.drop(columns=['positions'])
```

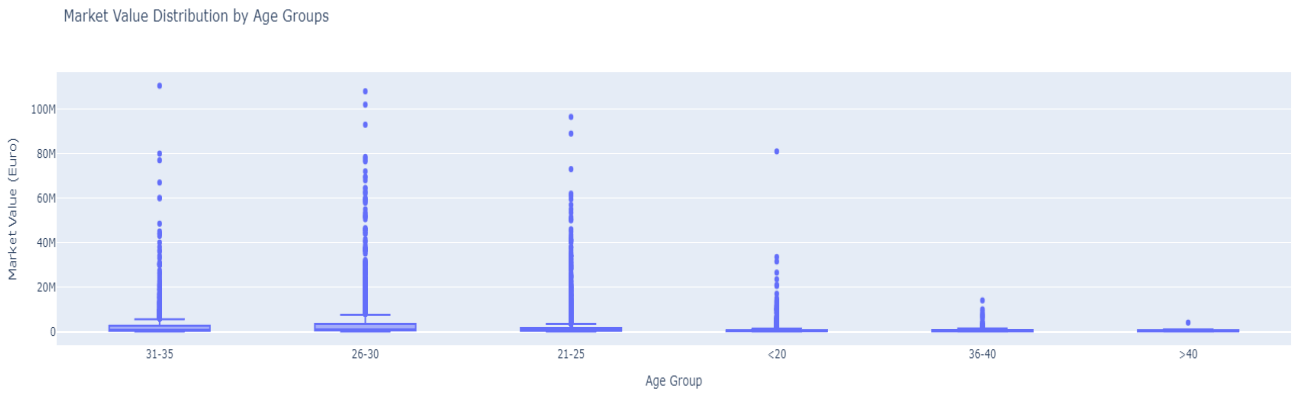
Veri ön işleme aşaması, modeli kötü etkileyen özellikler çıkarıldı ve metinsel verileri sayısal değerlere dönüştürdük.

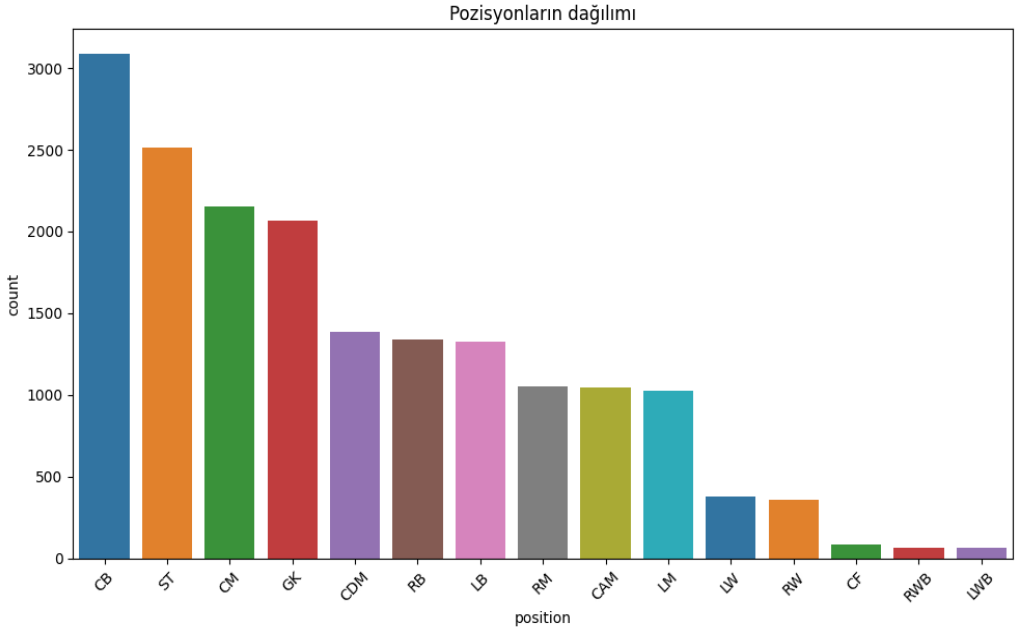


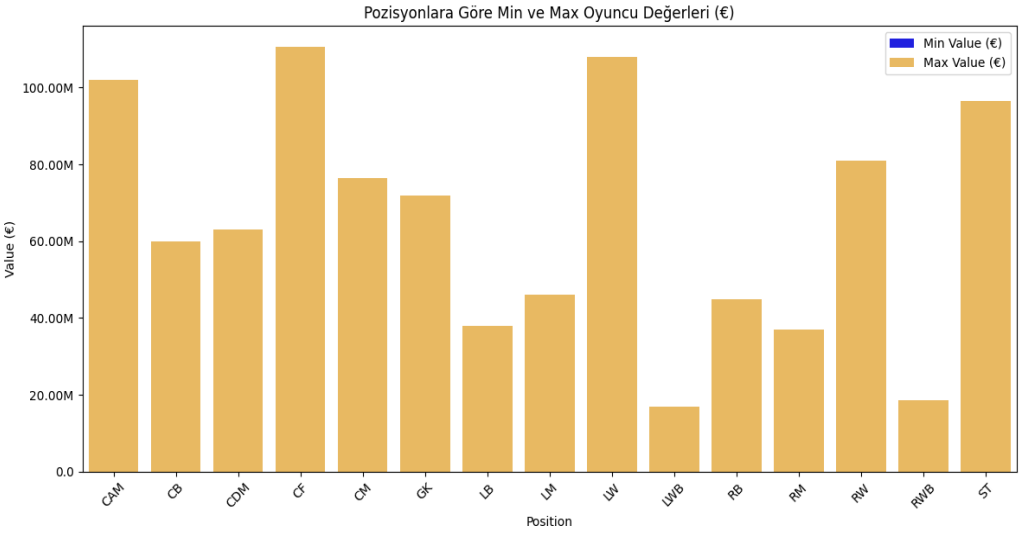
Veri setinde oyuncuların rating değerlerine göre dağılımı

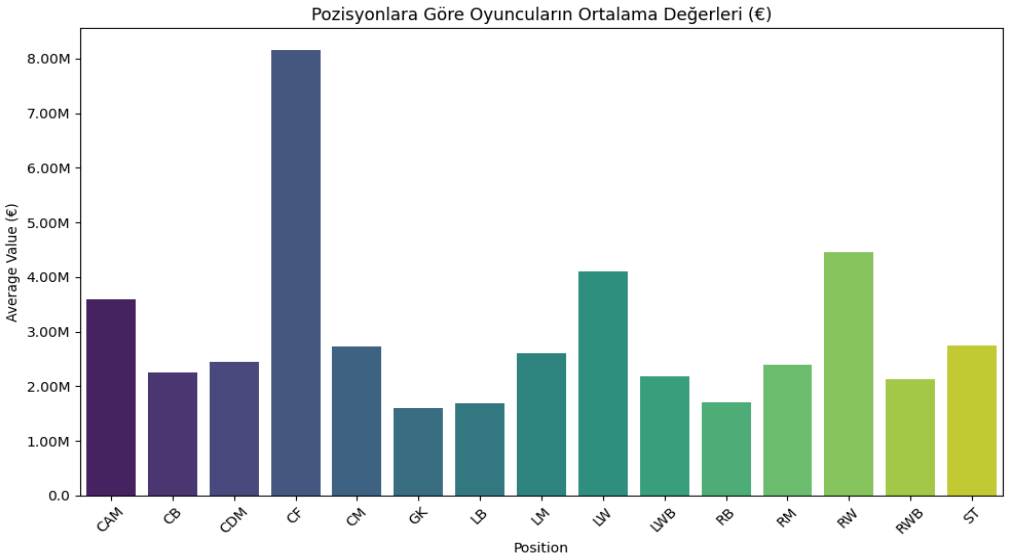


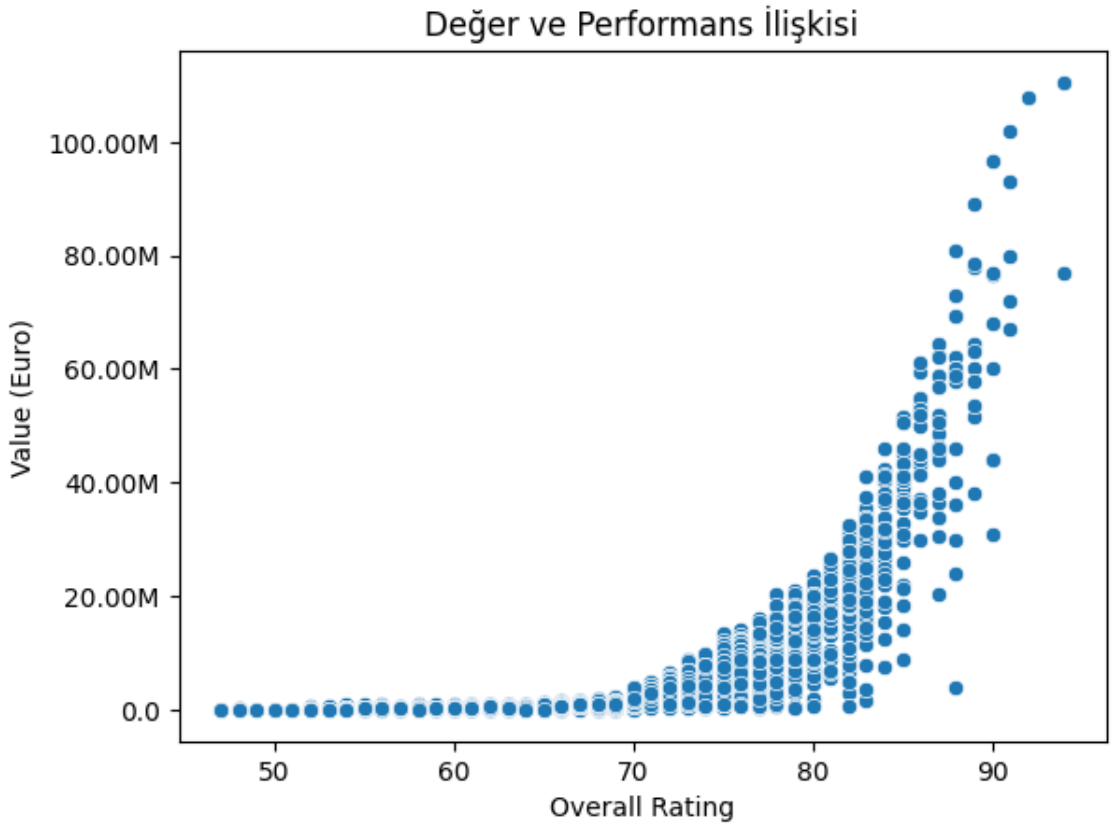
Veri setinde oyuncuların ülkelerine göre dağılımı

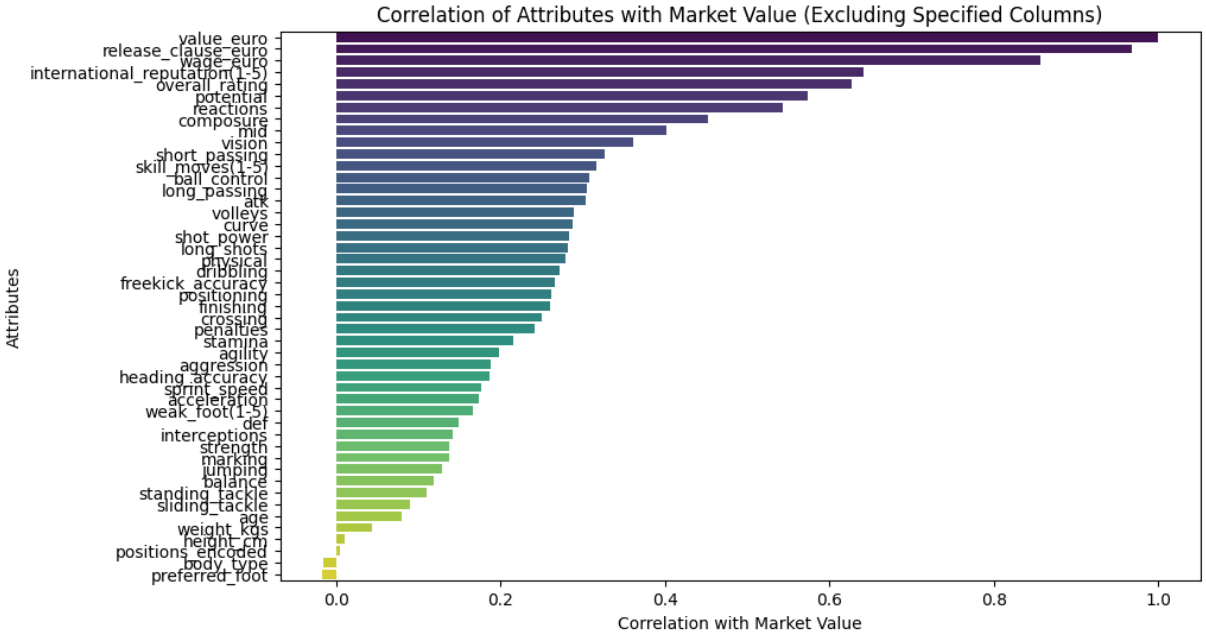



Pozisyonların dağılımı
count
position
CB
ST
CM
GK
CDM
RB
LB
RM
CAM
LM
LW
RW
CF
RWB
LWB


Pozisyonlara Göre Min ve Max Oyuncu Değerleri (€)
Min Value (€)
Max Value (€)
Value (€)
Position


Pozisyonlara Göre Oyuncuların Ortalama Değerleri (€)
Average Value (€)
Position


Değer ve Performans İlişkisi
Value (Euro)
100.00M
80.00M
60.00M
40.00M
20.00M
0.0
50
60
70
80
90
Overall Rating


Correlation of Attributes with Market Value (Excluding Specified Columns)
Attributes
value_euro
release_clause_euro
wage_euro
international_reputation(1-5)
overall_rating
potential
reactions
composure
mid
vision
short_passing
skill_moves(1-5)
ball_control
long_passing
att
volleys
curve
shot_power
long_shots
physical
dribbling
freekick_accuracy
positioning
finishing
crossing
penalties
stamina
agility
aggression
heading_accuracy
sprint_speed
acceleration
weak_foot(1-5)
def
interceptions
strength
marking
jumping
balance
standing_tackle
sliding_tackle
age
weight_kgs
height_cm
positions_encoded
body_type
preferred_foot
0.0
0.2
0.4
0.6
0.8
1.0
Correlation with Market Value

```
x = df.drop(columns=['value_euro','release_clause_euro','wage_euro'])


y = df['value_euro']
X_train,X_test,y_train,y_test=train_test_split(x,y,random_state=1,test_size=0.20)
```

```
from sklearn.feature_selection import mutual_info_regression
mutual_info = mutual_info_regression(x,y)
mutual_info = pd.Series(mutual_info, index=x.columns)
```

```
from sklearn.feature_selection import SelectPercentile
selected_top_columns = SelectPercentile(mutual_info_regression, percentile=20)
selected_top_columns.fit(X_train,y_train)
```

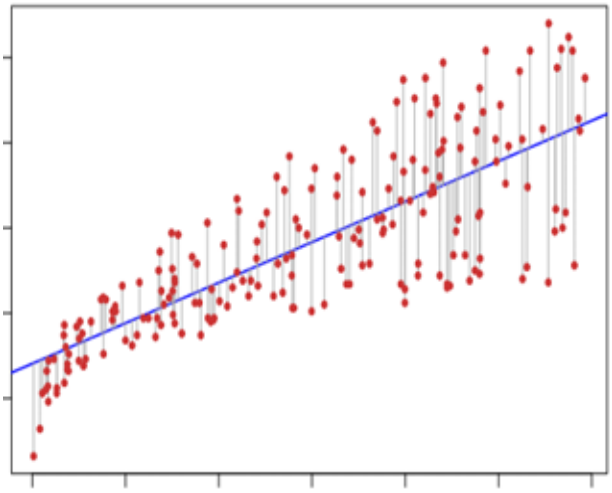
```
SelectPercentile(percentile=20,
                 score_func=<function mutual_info_regression at 0x000001FB80C7D260>)
```

```
X_train = pd.DataFrame(selected_top_columns.transform(X_train),
                       columns=X_train.columns[selected_top_columns.get_support()],
                       index=X_train.index)

X_test = pd.DataFrame(selected_top_columns.transform(X_test),
                      columns=X_test.columns[selected_top_columns.get_support()],
                      index=X_test.index)
```

Özellik seçimi için kullandığımız fonksiyon. Bu fonksiyon istediğimiz sonuç için en uygun özellikleri bizim için seçiyor.

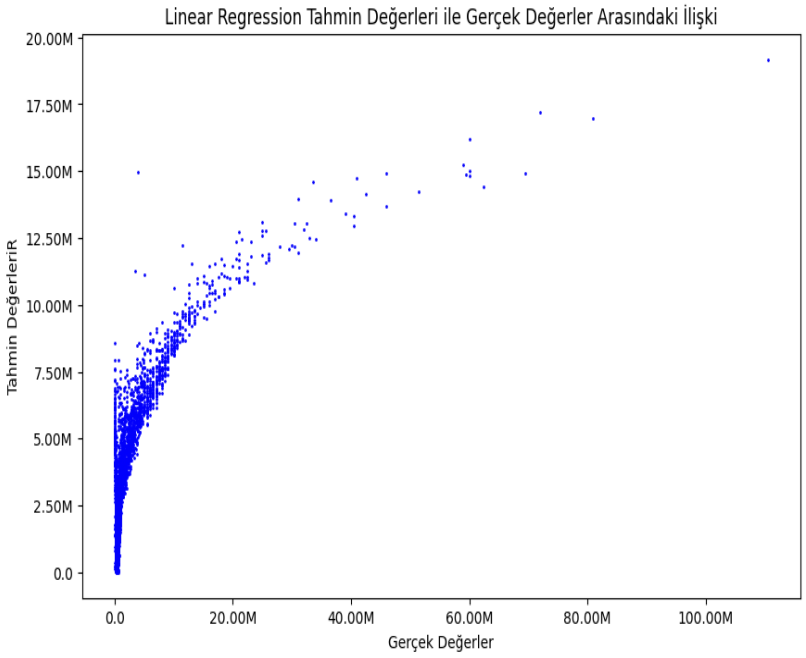
Doğrusal Regresyon

"Basit doğrusal regresyon analizi iki sürekli degişken arasındaki ilişkiyi nicelendiren istatistiksel bir tekniktir. Bu degişkenler, bağımlı degişken veya tahmin etmeye çalıştığınız degişken ve bağımsız veya tahmin edici degişken olarak ifade edilebilir. Tahmin edilen değerler ile gerçek değerler arasındaki hata değerini minimumda tutmak üzerine çalışır.



Doğrusal Regresyonun çalışma prensibi yukarıdaki şekilde gösterildiği gibidir.

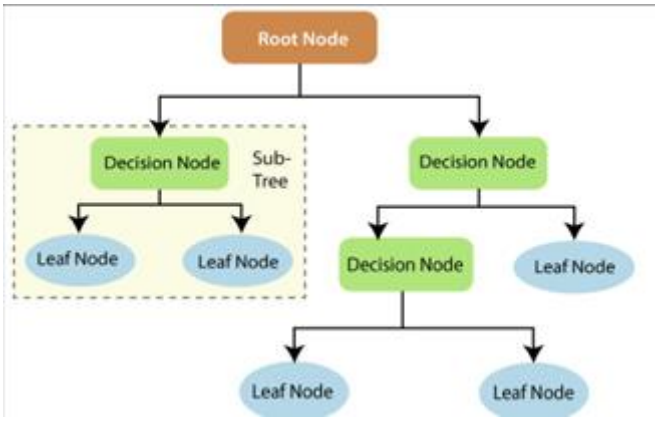
```python
reg_main = LinearRegression().fit(X=X_train, y=y_train)
reg_main_coef = reg_main.coef_
reg_main_intercept = reg_main.intercept_
reg_main_preds = reg_main.predict(X_test)
reg_main_pred_abs = abs(reg_main_preds)
reg_main_train_score = reg_main.score(X_train, y_train)
reg_main_test_score = reg_main.score(X_test, y_test)

reg_stats_df = pd.DataFrame({'PREDICT' : reg_main_preds, 'TEST' : y_test})
reg_stats_df['REMAINDER'] = reg_stats_df.apply(lambda x: x.PREDICT - x.TEST, axis=1)
reg_stats_df['REMAINDER_ABS'] = abs(reg_stats_df['REMAINDER'])

reg_main_score_train = reg_main.score(X_train, y_train)
reg_main_score_test = reg_main.score(X_test, y_test)

mse = mean_squared_error(y_test, reg_main_preds)
rmse = math.sqrt(mse)
mae = mean_absolute_error(y_test, reg_main_preds)
cross_val = cross_val_score(reg_main, X_train, y_train, cv=10)
r2 = r2_score(y_test, reg_main_preds)

lr_values = {'Linear Regression Preds'             : reg_main_preds,
             'Linear Regression Train Score'       : reg_main_train_score,
             'Linear Regression Test Score'        : reg_main_test_score,
             'Linear Regression Calculated Scores' : r2,
             'Linear Regression Stats DF'          : reg_stats_df}

plt.figure(figsize=(10,6))
plt.plot(y_test,reg_main_pred_abs,'ro',color='blue', markerfacecolor='red',markersize='1')
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(shorten_number))
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(shorten_number))
plt.title("Linear Regression Tahmin Değerleri ile Gerçek Değerler Arasındaki İlişki")
plt.xlabel("Gerçek Değerler")
plt.ylabel("Tahmin DeğerleriR")
plt.show()
```



Train Score	**0.4474**
Test Score	**0.4378**
Mean Absolute Error	**2231728.93**
Mean Squared Error	**18191540911558**
Root Mean Squared Error	**42641542658**
R2 Score	**0.4252**

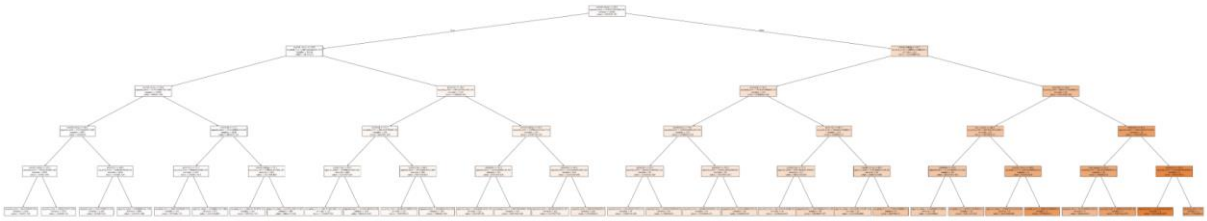
Karar Ağacı

Karar Ağacı isminden de anlaşılacağı üzere ağaç biçiminde bir yapı oluşturularak karar verme işleminin yapıldığı bir yapıdır. Karar ağaçlarında karar düğümleri (decision nodes) ve yaprak düğümleri (leaf nodes) bulunmaktadır. Karar düğümleri karar verme, sınıflandırma yapma, tahmin etme gibi işlemleri gerçekleştirirken yaprak düğümleri ise bu kararların tutulduğu yerdir. Ağacın bası kök düğüm (root node) olarak isimlendirilir ve karara bağlanmak için kök düğümden yaprak düğümlere doğru belli bir yol izlenmektedir



Karar Ağacı şeması yukarıdaki şekilde gösterildiği gibidir.

```
regressor = DecisionTreeRegressor(random_state=0, max_depth=5)
regressor.fit(X_train, y_train)
dtree_pred = regressor.predict(X_test)

dtree_pred_abs = abs(dtree_pred)
dtree_score_train = regressor.score(X_train, y_train)
dtree_score_test = regressor.score(X_test, y_test)



mse = mean_squared_error(y_test, dtree_pred)
rmse = math.sqrt(mse)
mae = mean_absolute_error(y_test, dtree_pred)
cross_val = cross_val_score(regressor, X_train, y_train, cv=10)
r2 = r2_score(y_test, dtree_pred)
dtree_values = {'Mean Squared Error'       :   mse,
                'Root Mean Squared Error'  :   rmse,
                'Mean Absolute Error'      :   mae,
                'Cross Validation'         :   cross_val,
                'R2 Score'                 :   r2}

fig = plt.figure(figsize=(100,20))
_ = tree.plot_tree(regressor, feature_names=X_train.columns,filled=True)
fig.savefig("decistion_tree.png")



print(dtree_values)
```



Bizim modelimizde karar ağacı veriyi bu şekilde böldü. Root olarak rating değeri belirlendi.

Train Score	**0.9526**
Test Score	**0.9214**
Mean Absolute Error	**529846.60**
Mean Squared Error	**2541524994969**
Root Mean Squared Error	**1594216205**
R2 Score	**0.9214**

Rastgele Orman

Random Forest, birden fazla karar ağacının (decision tree) bir araya getirilmesiyle oluşan bir topluluk (ensemble) yöntemidir. Her bir ağaç bağımsız olarak eğitilir ve sonuçları çoğunluk oylaması (sınıflandırma) veya ortalama (regresyon) ile birleştirilir.

```
forest_regressor = RandomForestRegressor(n_estimators=100,random_state=42)
forest_regressor.fit(X_train, y_train)
forest_pred = forest_regressor.predict(X_test)

forest_pred_abs = abs(forest_pred)
forest_score_train = forest_regressor.score(X_train, y_train)
forest_score_test = forest_regressor.score(X_test, y_test)

print(forest_score_test)
print(forest_score_train)

mse = mean_squared_error(y_test, forest_pred)
rmse = math.sqrt(mse)
mae = mean_absolute_error(y_test, forest_pred)
cross_val = cross_val_score(forest_regressor, X_train, y_train, cv=10)
r2 = r2_score(y_test, forest_pred)
forest_values = {'Mean Squared Error'       :   mse,
                 'Root Mean Squared Error'  :   rmse,
                 'Mean Absolute Error'      :   mae,
                 'Cross Validation'         :   cross_val,
                 'R2 Score'                 :   r2}


print(forest_values)
```

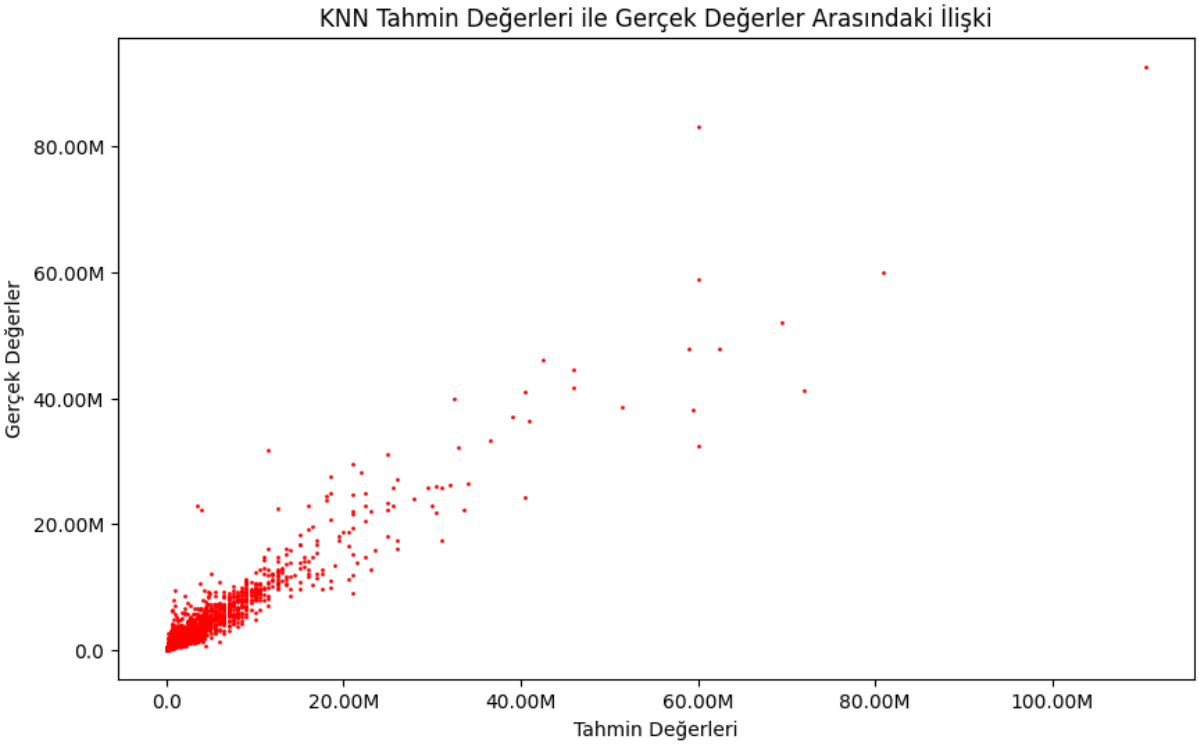
Train Score	**0.9944**
Test Score	**0.9545**
Mean Absolute Error	**310695.32**
Mean Squared Error	**1237790307140**
Root Mean Squared Error	**1112560248**
R2 Score	**0.9545**

K-En Yakın Komşu Algoritması

Sınıflandırma ve regresyon işlemlerinde kullanılabilen ve sıklıkla tercih edilen algoritmalardan biridir. Özniteliklerin birbirlerine olan uzaklıklarının Öklid işlemi ile hesaplanması ile isminde de geçen K parametresi ile en yakın K sayıdaki örneğe bakarak sınıflandırma işleminin yapıldığı algoritmadır.

Algoritmamızda en uygun k değeri 3 olarak belirlenmiştir. Bu değer bulunurken 1-40 arasındaki değerler gridsearch ile denenmiş olup en iyi sonuç alınmıştır.

```
# KNN
# En iyi K değerinin belirlenmesi
params = {'n_neighbors' : range(40)}
knn = KNeighborsRegressor()
model = GridSearchCV(knn, params, cv=8)
model.fit(X_train,y_train)
print("BEST: ",model.best_params_)

# Belirlenen en iyi K değeri için KNN algoritmasının oluşturulması
best_knn = KNeighborsRegressor(n_neighbors=model.best_params_['n_neighbors'])
#best_knn = KNeighborsRegressor(n_neighbors=5)
best_knn.fit(X_train, y_train)
best_knn_pred = best_knn.predict(X_test)
best_knn_pred_abs = abs(best_knn_pred)



best_knn_score_train = best_knn.score(X_train, y_train)
best_knn_score_test = best_knn.score(X_test, y_test)

mse = mean_squared_error(y_test, best_knn_pred)
rmse = math.sqrt(mse)
mae = mean_absolute_error(y_test, best_knn_pred)
cross_val = cross_val_score(best_knn, X_train, y_train, cv=10)
r2 = r2_score(y_test, best_knn_pred)
knn_best_values = {'Mean Squared Error'       :   mse,
                   'Root Mean Squared Error'  :   rmse,
                   'Mean Absolute Error'      :   mae,
                   'Cross Validation'         :   cross_val,
                   'R2 Score'                 :   r2}
plt.figure(figsize=(10,6))
plt.plot(y_test,best_knn_pred_abs,'ro',color='red', markerfacecolor='red',markersize='1')
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(shorten_number))
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(shorten_number))
plt.title("KNN Tahmin Değerleri ile Gerçek Değerler Arasındaki İlişki")
plt.xlabel("Tahmin Değerleri")
plt.ylabel("Gerçek Değerler")
plt.show()

print(knn_best_values)
```

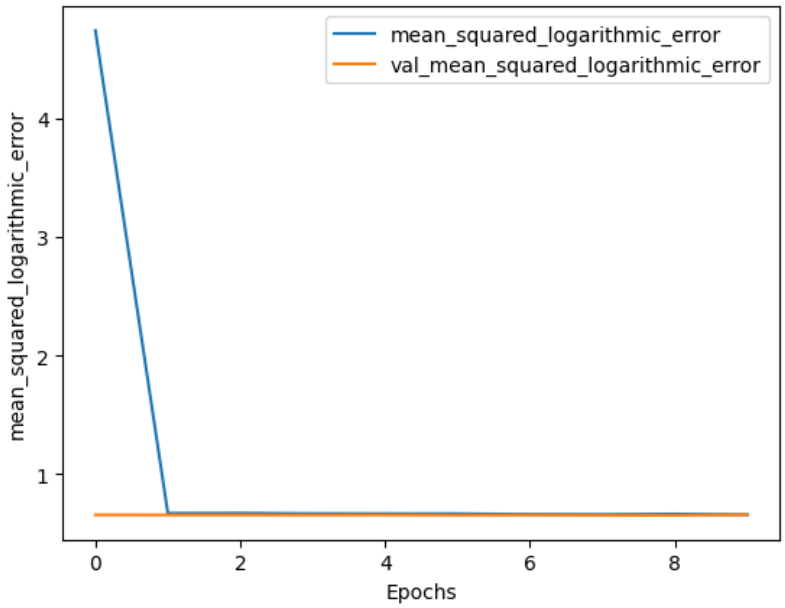


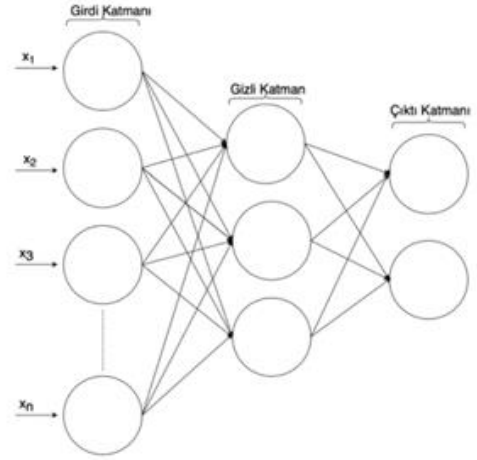
Train Score	**0.9334**
Test Score	**0.9134**
Mean Absolute Error	**584202.64**
Mean Squared Error	**3097582801192**
Root Mean Squared Error	**1814199106**
R2 Score	**0.9134**

Yapay Sinir Ağları

Biyolojik sinir sistemlerinden esinlenerek geliştirilen ve özellikle karmaşık veri setlerinde yüksek doğrulukla tahmin yapabilen güçlü bir makine öğrenmesi algoritmasıdır. Sinir ağları, çok sayıda bağlantılı yapay nörondan oluşur ve bu nöronlar arasındaki bağlantılar öğrenme süreci boyunca ayarlanır.Yapay Sinir Ağları, verileri katmanlar halinde işleyen ve her bir katmanın çıkışını bir sonraki katmana giriş olarak veren çok katmanlı bir yapı içerir.

```
def build_model_using_sequential():
    model = Sequential([
    Dense(120, kernel_initializer='normal', activation='relu'),
    Dropout(0.2),
    Dense(60, kernel_initializer='normal', activation='relu'),
    Dropout(0.2),
    Dense(30, kernel_initializer='normal', activation='relu'),
    Dense(1, kernel_initializer='normal', activation='linear')
    ])
    return model
model = build_model_using_sequential()
msle = MeanSquaredLogarithmicError()
model.compile(loss=msle,optimizer=Adam(learning_rate=0.01),metrics=[msle]
)
history = model.fit(
    X_train.values, y_train.values, epochs=10, batch_size=64, validation_split=0.2
)
def plot_history(history, key):
    plt.plot(history.history[key])
    plt.plot(history.history['val_'+key])
    plt.xlabel("Epochs")
    plt.ylabel(key)
    plt.legend([key, 'val_'+key])
    plt.show()
plot_history(history, 'mean_squared_logarithmic_error')

neural_preds = model.predict(X_test)
neural_preds_ed = []
for i in neural_preds:
    for x in i:
        neural_preds_ed.append(x)
neural_preds_ed2 = pd.Series(neural_preds_ed)

neural_df = pd.DataFrame({'PREDICT' : neural_preds_ed, 'TEST' : y_test})
neural_df['REMAINDER'] = neural_df.apply(lambda x: x.PREDICT- x.TEST, axis=1)
neural_df['REMAINDER_ABS'] = abs(neural_df['REMAINDER'])

neural_mse = mean_squared_error(y_test, neural_preds_ed)
neural_rmse = math.sqrt(neural_mse)
neural_mae = mean_absolute_error(y_test, neural_preds_ed)
neural_r2 = r2_score(y_test, neural_preds_ed)
```





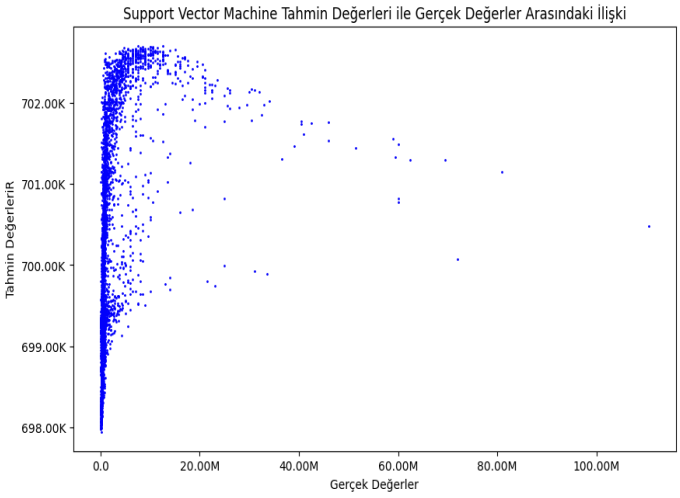
Yapay sinir ağları çalışma prensibi şekildeki gibidir.

Train Score	**0.7122**
Test Score	**0.6984**
Mean Absolute Error	**1200704.54**
Mean Squared Error	**9759129786205**
Root Mean Squared Error	**3123960592**
R2 Score	**0.6984**

Destek Vektör Makineleri

Denetimli öğrenme algoritmalarından biri olup, genellikle sınıflandırma ve regresyon problemlerinde kullanılır. SVM, verileri sınıflandırırken, veriler arasındaki sınıf sınırlarını en iyi şekilde belirlemeye çalışır. Bu algoritma, veri noktalarını daha iyi ayırabilmek için, iki sınıf arasındaki en geniş marjini (mesafeyi) bulmayı hedefler.

```
svr_regressor = SVR(kernel='rbf',C=1) # default
svr_regressor.fit(X_train, y_train)
svr_preds = svr_regressor.predict(X_test)
svr_preds_abs = abs(svr_preds)
svr_train_score = svr_regressor.score(X_train, y_train)
svr_test_score = svr_regressor.score(X_test, y_test)

mse = mean_squared_error(y_test, svr_preds)
rmse = math.sqrt(mse)
mae = mean_absolute_error(y_test, svr_preds)
cross_val = cross_val_score(svr_regressor, X_train, y_train, cv=10)
r2 = r2_score(y_test, svr_preds)
svr_values = {'Mean Squared Error'       :   mse,
              'Root Mean Squared Error'  :   rmse,
              'Mean Absolute Error'      :   mae,
              'Cross Validation'         :   cross_val,
              'R2 Score'                 :   r2}


plt.figure(figsize=(10,6))
plt.plot(y_test,svr_preds_abs,'ro',color='blue', markerfacecolor='red',markersize='1')
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(shorten_number))
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(shorten_number))
plt.title("Support Vector Machine Tahmin Değerleri ile Gerçek Değerler Arasındaki İlişki")
plt.xlabel("Gerçek Değerler")
plt.ylabel("Tahmin DeğerleriR")
plt.show()

print(svr_values)
```



Train Score	**0.0941**
Test Score	**0.0966**
Mean Absolute Error	**2104271.14**
Mean Squared Error	**35408663968335**
Root Mean Squared Error	**58505179579**
R2 Score	**0.0966**

Sonuç ve Tartışma

	R2 Score
Doğrusal Regresyon	0.4552
Karar Ağacı	0.9214
Rastgele Orman	0.9545
K-En Yakın Komşu	0.9134
Yapay Sinir Ağları	0.6984
Destek Vektör Makineleri	0.0966

Alınan sonuçlara göre en iyi sonucu veren Rastgele orman en kötü sonucu veren Destek Vektör Makineleri algoritması olmuştur.

Algoritmaların overfit olup olmadığını anlamak amacıyla kfold kütüphanesi kullanılmıştır.

Ayrıca model veri seti dışındaki veriler ile de test edilmiştir.

Projenin ilerleyen aşamalarında veri setine bağımlı kalınmayıp API kullanılarak istenilen futbolcunun değerinin bulunması planlanmaktadır.