Ev Fiyatları Tahmini için Makine Öğrenmesi Uygulaması

Giriş

Ev fiyatları, hem alıcılar hem de satıcılar için büyük bir öneme sahip olan karmaşık bir konudur. Ev alıcıları, doğru fiyatı belirleyerek bütçelerine uygun bir ev bulmak isterken, satıcılar da evlerini en iyi fiyata satmak isterler. Makine öğrenmesi teknikleri, ev fiyatlarını tahmin etmek için güçlü bir araçlar sunar ve bu sunumda, bir evin fiyatını tahmin etmek için nasıl bir makine öğrenmesi modeli oluşturabileceğimizi inceleyeceğiz.

Problem Tanımı

1. **Yeni Ev Alımı:** 232 metrekarelik bir ev bakıyoruz ve fiyatının ne olacağını merak ediyoruz.
2. **Ev Satışı:** Mevcut bir evimiz var ve en uygun fiyatı belirlemek istiyoruz.

Veri Seti Seçimi

Ev fiyatlarını tahmin etmek için kullanabileceğimiz çeşitli veri setleri bulunmaktadır. Bu sunumda, ev fiyatlarını tahmin etmek için yaygın olarak kullanılan ve açık kaynaklı bir veri seti olan "House Prices: Advanced Regression Techniques" veri setini kullanacağız.

Adımlar

1. **Veri Keşfi:** Veri setini inceleyerek ev fiyatlarını etkileyen faktörleri anlamak.
2. **Veri Ön İşleme:** Eksik verileri doldurma, kategorik özellikleri kodlama, özellik mühendisliği gibi işlemleri gerçekleştirmek.
3. **Model Oluşturma:** Lineer regresyon gibi bir model seçerek modeli oluşturmak.
4. **Model Eğitimi:** Veri setini kullanarak modeli eğitmek.
5. **Model Değerlendirmesi:** Modelin performansını değerlendirmek ve sonuçları yorumlamak.
6. **Tahminler:** Belirlenen bir ev büyüklüğü için fiyat tahminlerini yapmak.

Sonuç

Makine öğrenmesi teknikleri, ev fiyatlarını tahmin etmek için güçlü bir araçlar sunar. Bu sunumda, ev fiyatlarını tahmin etmek için nasıl bir makine öğrenmesi modeli oluşturabileceğimizi inceledik. Ev fiyatları tahmini, ev alıcıları ve satıcıları için değerli bir bilgidir ve doğru veri ve model kullanımıyla bu tahminlerin doğruluğu artırılabilir.

1.Veri Keşfi

Veri keşfi aşamasında, "House Prices: Advanced Regression Techniques" veri setini inceleyerek ev fiyatlarını etkileyen faktörleri anlamaya çalışacağız. Bu adımı gerçekleştirmek için Python programlama dilinde pandas ve matplotlib gibi kütüphanelerden yararlanabiliriz.

```
import pandas as pd
import matplotlib.pyplot as plt

# Veri setini yükleme
df = pd.read_csv('house_prices.csv')

# Veri setinin ilk birkaç satırını inceleme
print(df.head())

# Veri setinin genel istatistiklerini görüntüleme
print(df.describe())

# Veri setindeki özelliklerin dağılımlarını görselleştirme
df.hist(bins=20, figsize=(15, 10))
plt.show()
```

Bu kod parçacığı, "house_prices.csv" dosyasından veri setini yükler, ilk birkaç satırını ve genel istatistiklerini görüntüler, ve veri setindeki özelliklerin dağılımlarını histogramlar aracılığıyla görselleştirir.

Veri setini daha ayrıntılı bir şekilde inceledikten sonra, ev fiyatlarını etkileyen faktörleri anlamak için özellikler arasındaki ilişkileri ve dağılımları daha derinlemesine inceleyebiliriz.

3. Adım: Veri Temizliği ve Ön İşleme

Veri temizliği ve ön işleme aşamasında, veri setindeki eksik değerleri ele alacak, gereksiz veya tekrarlayan verileri kaldıracağız ve kategorik özellikleri sayısal değerlere dönüştüreceğiz. Bu adımda, veri setimizi modelimizin kullanabileceği uygun bir formata getireceğiz.

```python
# Eksik değerleri kontrol etme
print(df.isnull().sum())

# Gereksiz veya tekrarlayan verileri kaldırma (opsiyonel)
# df = df.drop(columns=['gereksiz_özellik_1', 'gereksiz_özellik_2'])

# Kategorik özellikleri sayısal değerlere dönüştürme
df = pd.get_dummies(df, drop_first=True)

# Özellikler arasındaki korelasyonları inceleme
correlation_matrix = df.corr()
print(correlation_matrix)
```

Bu kod parçacığı, veri setindeki eksik değerleri kontrol eder, gereksiz veya tekrarlayan verileri kaldırabilir (bu adım opsiyoneldir), kategorik özellikleri sayısal değerlere dönüştürür ve özellikler arasındaki korelasyonu inceleyerek ilişkileri anlamamıza yardımcı olur.

3. Adım: Verilerin Eğitim ve Test Setlerine Ayrılması

Makine öğrenimi modelimizi eğitmek ve test etmek için veri setimizi eğitim ve test setlerine böleceğiz. Bu, modelin gerçek veriler üzerinde nasıl performans gösterdiğini değerlendirmemize olanak sağlar.

```python
from sklearn.model_selection import train_test_split

# Bağımsız değişkenler (özellikler)
X = df.drop(columns=['fiyat'])

# Bağımlı değişken (hedef)
y = df['fiyat']

# Veri setini eğitim ve test setlerine ayırma
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Eğitim ve test setlerinin boyutunu kontrol etme
print("Eğitim seti boyutu:", X_train.shape)
print("Test seti boyutu:", X_test.shape)
```

Bu kod parçacığı, veri setini bağımlı ve bağımsız değişkenlere ayırır, sonra bu verileri eğitim ve test setlerine böler. Bu işlem, genellikle verinin %80'ini eğitim setine ve %20'sini test setine ayırarak gerçekleştirilir.

---

4. Adım: Model Oluşturma ve Eğitme

Şimdi, veri setimizi kullanarak makine öğrenimi modelimizi oluşturup eğiteceğiz. Bu adımda, belirlediğimiz bir algoritmayı kullanarak modeli eğiteceğiz.

```
from sklearn.linear_model import LinearRegression

# Lineer regresyon modeli oluşturma
model = LinearRegression()

# Modeli eğitme
model.fit(X_train, y_train)
```

Bu kod parçacığı, **LinearRegression** sınıfından bir model oluşturur ve daha sonra **fit** yöntemiyle eğitim seti üzerinde bu modeli eğitir. Model, bağımlı değişkeni (fiyat) belirli bağımsız değişkenlere (ev özellikleri) dayanarak tahmin etmeyi öğrenir.

5. Adım: Modeli Test Etme

Modelimizi eğittikten sonra, modelin performansını değerlendirmek için test seti üzerinde test etmeliyiz. Bu adımda, modelimizin test verileri üzerinde ne kadar iyi performans gösterdiğini görmek için bir dizi değerlendirme metriği kullanabiliriz

```
from sklearn.metrics import mean_squared_error

# Test verileri üzerinde tahmin yapma
y_pred = model.predict(X_test)

# Modelin performansını değerlendirme
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Bu kod parçacığı, modelimizin test verileri üzerindeki tahminlerini oluşturur ve ardından gerçek değerlerle karşılaştırarak bir hata ölçüsü olan ortalama karesel hatayı hesaplar. Daha düşük bir ortalama karesel hata değeri, modelin daha iyi performans gösterdiğini gösterir.

6. Adım: Sonuçları Görselleştirme

Modelimizin performansını görselleştirmek, sonuçları daha anlaşılır hale getirebilir ve modelin davranışını daha iyi anlamamıza yardımcı olabilir. Bu adımda, genellikle modelin tahminlerini gerçek değerlerle karşılaştıran grafikler oluştururuz.

```
import matplotlib.pyplot as plt

# Scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Gerçek Değerler")
plt.ylabel("Tahminler")
plt.title("Gerçek Değerler vs. Tahminler")
plt.show()
```

Bu kod parçacığı, gerçek değerlerle tahmin edilen değerler arasındaki ilişkiyi gösteren bir scatter plot oluşturur. Eğer modelimiz iyi çalışıyorsa, scatter plot'taki noktaların bir doğruya yakın bir şekilde sıralanması beklenir.