Weekly ve Auto Veri Seti üzerine modellemeler

Alican Tunç
Veri Bilimi ve Büyük Veri Yüksek Lisans Programı
Yıldız Teknik Üniversitesi
Istanbul, TÜRKİYE:
can.tunc1@std.yildiz.edu.tr

***Abstract*—Bu çalışmada, Weekly ve Auto veri setleri kullanılarak çeşitli sınıflandırma algoritmaları adım adım uygulanmış ve performansları karşılaştırılmıştır. Weekly veri seti üzerinde lojistik regresyon, K-En Yakın Komşu (KNN) ve Naive Bayes yöntemleri denenmiş, eğitim ve test verileri ayrılarak doğruluk analizleri yapılmıştır. Auto veri setinde ise destek vektör makineleri (SVM) farklı çekirdek fonksiyonları ve parametrelerle modellenmiş, 5 katlı çapraz doğrulama ile doğruluk skorları karşılaştırılmıştır. Elde edilen sonuçlar, farklı yöntemlerin ve parametre ayarlarının sınıflandırma başarısına etkisini ortaya koymakta, model seçiminde rehberlik etmektedir. Çalışma, temel makine öğrenimi tekniklerinin gerçek veri setleri üzerinde uygulanabilirliğini ve performans değerlendirmesini sunmaktadır.**

Keywords—Weekly veri seti, Auto veri seti, lojistik regresyon, K-En Yakın Komşu, Naive Bayes, destek vektör makineleri, sınıflandırma, çapraz doğrulama.,

I. *GIRIŞ*

Bu çalışmada, finansal ve otomotiv alanlarına ait iki farklı veri seti kullanılarak makine öğrenmesi ve istatistiksel modelleme yöntemlerinin uygulaması incelenmiştir. Weekly veri seti, 1990-2010 yılları arasındaki haftalık hisse senedi getirilerini içerirken, Auto veri seti araçların yakıt verimliliği ve özelliklerine dair bilgileri barındırmaktadır.

Weekly veri seti üzerinde lojistik regresyon, K-En Yakın Komşu (KNN) ve Naive Bayes sınıflandırma yöntemleri uygulanmış, farklı değişken kombinasyonları ve veri bölünmeleri ile modellerin performansları karşılaştırılmıştır. Model doğrulukları karışıklık matrisi ve sınıflandırma oranları ile değerlendirilmiştir.

Auto veri setinde ise, araçların yakıt verimliliğinin yüksek veya düşük olarak sınıflandırılması amaçlanmış; destek vektör makineleri (SVM) algoritması kullanılarak farklı parametre ve çekirdek fonksiyonları ile modeller oluşturulmuş, 5 katlı çapraz doğrulama ile model doğrulukları karşılaştırılmıştır.

Çalışmada Python ve R programlama dilleri etkin şekilde kullanılarak, veri ön işleme, model kurma, değerlendirme ve sonuçların görselleştirilmesi adımları gerçekleştirilmiştir. Elde edilen bulgular, farklı veri setlerinde uygulanan makine öğrenmesi yöntemlerinin sınıflandırma başarısını ve modelleme yaklaşımlarının etkinliğini ortaya koymaktadır.

II. *DENEYSEL ANALİZ VE YORUMLAMA*

A. Veri Seti

Bu çalışmada iki farklı veri seti kullanılmıştır: **Weekly** ve **Auto**.

Weekly veri seti, 1990 yılından 2010 yılına kadar haftalık hisse senedi getirilerine ilişkin 1089 gözlem içermektedir. Veri seti, finansal piyasaların haftalık yönünü (Direction) belirlemek amacıyla kullanılmıştır. Açıklayıcı değişkenler olarak önceki haftaların getiri değerleri (Lag1, Lag2, Lag3, Lag4, Lag5) ve işlem hacmi (Volume) yer almaktadır. Bu veri seti, ikili sınıflandırma problemleri için uygun olup, finansal zaman serisi analizinde uygulanacak makine öğrenmesi algoritmalarının performansını değerlendirmek amacıyla seçilmiştir.

Auto veri seti ise, araçların yakıt verimliliği (mpg) ve çeşitli özelliklerine dair bilgiler içermektedir. Toplam 392 araç gözlemi bulunmaktadır. Bu veri setinde, araçların yakıt verimliliğinin yüksek veya düşük olması ikili sınıflandırma problemi olarak ele alınmıştır. Medyan mpg değerine göre sınıflandırılan hedef değişken kullanılarak destek vektör makineleri (SVM) modelleri geliştirilmiş ve farklı parametre ayarlarıyla performansları karşılaştırılmıştır.

Bu iki veri seti, farklı problem türleri ve alanlar için seçilmiş olup, sınıflandırma yöntemlerinin genel geçer başarısını ölçmek için kullanılmıştır.

B. Yöntem

2.1. Weekly Veri Seti için:

- **Lojistik Regresyon, KNN ve Naive Bayes** yöntemleri uygulanmıştır.
- Weekly veri setindeki gözlemler eğitim ve test olarak zaman serisine uygun şekilde bölünmüş, 1990-2008 yılları eğitim, 2009-2010 yılları test olarak kullanılmıştır.
- Lojistik regresyon modelinde Direction bağımlı değişken, Lag1, Lag2, Lag3, Lag4, Lag5 ve Volume bağımsız değişkenler olarak kullanılmıştır. Modelde anlamlı değişkenler belirlenmiş ve yorumlanmıştır.
- KNN algoritması farklı K değerleri (komşu sayıları) ile test edilmiş, en uygun K doğruluk oranları ile seçilmiştir.
- Naive Bayes sınıflandırıcısı da aynı eğitim-test ayrımı ile uygulanarak karşılaştırılmıştır.
- Modellerin performansı karışıklık matrisleri ve doğru sınıflandırma oranları ile değerlendirilmiştir
- Veri seti .csv ile değil kullanımı daha kolay olması adına ilgili kütüphanelerden ve github repolarından çekilmiştir.

2.2 Auto Veri Seti için:

- Medyan mpg değerine göre ikili hedef değişken oluşturulmuş ve destek vektör makineleri (SVM) uygulanmıştır.
- Sayısal bağımsız değişkenler kullanılmış, mpg hedef değişken açıklayıcı değişken olarak dahil edilmemiştir.
- Lineer, Radial Basis Function (RBF) ve polinom çekirdek fonksiyonları farklı parametrelerle (C, gamma, degree) kullanılarak 5 katlı çapraz doğrulama gerçekleştirilmiştir.
- Model doğrulukları karşılaştırılmış ve parametre etkileri grafiksel olarak incelenmiştir.
- Veri seti .csv ile değil kullanımı daha kolay olması adına ilgili kütüphanelerden ve github repolarından çekilmiştir.

C. *2.2.1 Weekly veri seti için İşlemler*

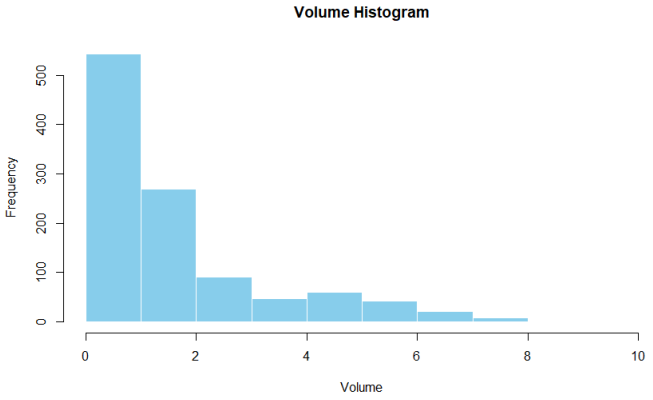
Weekly için, R kullanarak aşağıdaki şekilde tanımlayıp modelimizi kuruyoruz.

```
library(ISLR)
data(Weekly)
summary(Weekly)

# Histogram: Volume
hist(Weekly$Volume, main="Volume Histogram", xlab="Volume", col="skyblue", border="white")

# Boxplot: Lag değişkenleri
boxplot(Weekly[,2:6], main="Lag Değişkenleri Boxplot")

plot(Weekly$Volume, type="l", main="Volume Zaman Serisi", ylab="Volume", xlab="Hafta")

pairs(Weekly[,2:6], col=ifelse(Weekly$Direction=="Up", "green", "red"), main="Lag Değişkenleri ve Yön")

# Lojistik regresyon modeli (binomial aile)
model <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
             data = Weekly,
             family = binomial)
```

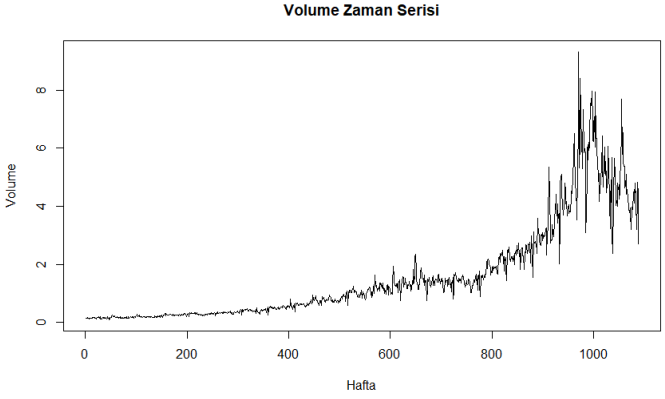
Şekil 1: Weekly modelinin oluşturulması

```
> summary(Weekly)
      Year          Lag1               Lag2               Lag3               Lag4               Lag5         
 Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580   1st Qu.: -1.1580   1st Qu.: -1.1660  
 Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410   Median :  0.2380   Median :  0.2340  
 Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472   Mean   :  0.1458   Mean   :  0.1399  
 3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4050  
 Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260  
     Volume           Today          Direction  
 Min.   :0.08747   Min.   :-18.1950   Down:484  
 1st Qu.:0.33202   1st Qu.: -1.1540   Up  :605  
 Median :1.00268   Median :  0.2410             
 Mean   :1.57462   Mean   :  0.1499             
 3rd Qu.:2.05373   3rd Qu.:  1.4050             
 Max.   :9.32821   Max.   : 12.0260             
```

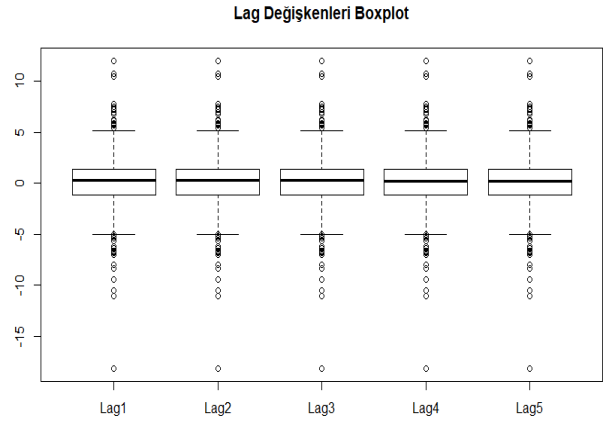
Şekil 2: Weekly modelinin özeti



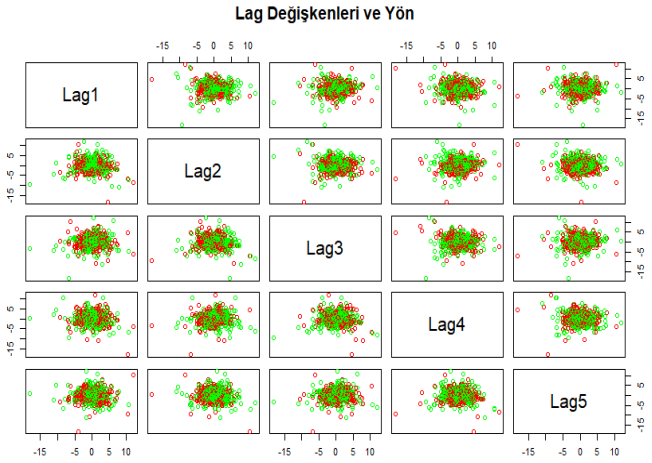
Şekil 3:Volume histogramı



Şekil 4: Volume zaman serisi



Şekil 5: Lag değişkenleri boxplot ile



Şekil 6: Lag ve direction ilişkisi

```
> cor(Weekly[-9])
              Year         Lag1         Lag2         Lag3         Lag4         Lag5       Volume        Today
Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923 -0.030519101  0.84194162 -0.032459894
Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876 -0.008183096 -0.06495131 -0.075031842
Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535 -0.072499482 -0.08551314  0.059166717
Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865  0.060657175 -0.06928771 -0.071243639
Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000 -0.075675027 -0.06107462 -0.007825873
Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027  1.000000000 -0.05851741  0.011012698
Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617 -0.058517414  1.00000000 -0.033077783
Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873  0.011012698 -0.03307778  1.000000000
```

Şekil 7: Korelasyon matrisi

Başlangıçtaki tanımlayıcı istatistikler ve serpilme grafikleri incelendiğinde, belirgin bir örüntü gözlemlenmemiştir. Ancak, haftalık işlem hacminin 1990'dan 2010'a kadar önemli ölçüde arttığı fark edilmiştir. Zaman içindeki işlem hacmi grafiğine bakıldığında, işlem hacminin 21 yıl boyunca katlanarak büyüdüğü açıkça görülmektedir. Değişkenler arasındaki korelasyon matrisine bakıldığında ise, lag (gecikmeli) değişkenlerin bugünkü getirilerle oldukça zayıf bir ilişkiye sahip olduğu anlaşılmıştır. Dikkat çeken tek yüksek korelasyon ise 0.84 ile Hacim (Volume) ve Yıl (Year) değişkenleri arasında olup, önceki grafikte gözlemlenen artışla uyumludur. Özetle haftalık işlem hacmi zamanla hızla artarken, diğer gecikmeli getiri değişkenlerinin bugünkü getiri ile ilişkisi zayıftır.

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6949  -1.2565   0.9913   1.0849   1.4579

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1        -0.04127    0.02641  -1.563   0.1181
Lag2         0.05844    0.02686   2.175   0.0296 *
Lag3        -0.01606    0.02666  -0.602   0.5469
Lag4        -0.02779    0.02646  -1.050   0.2937
Lag5        -0.01447    0.02638  -0.549   0.5833
Volume      -0.02274    0.03690  -0.616   0.5377
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.4  on 1082  degrees of freedom
AIC: 1500.4
```

Şekil 8: İlgili data setin farklı modellere göre acc değerleri

İstatistiksel olarak anlamlı tek değişken **Lag2**'dir. p-değeri 0.0296 olup, %5 anlamlılık düzeyinde, Lag2'nin **Direction** (yön) değişkeniyle ilişkisiz olduğu yönündeki sıfır hipotezini reddetmek için yeterli kanıt sunmaktadır. Diğer açıklayıcı değişkenlerin hiçbiri istatistiksel olarak anlamlı değildir. Ancak **Lag1** değişkeni, p-değeri 0.1181 ile %10 anlamlılık sınırına yakın olup, kısmen anlamlı sayılabilecek düzeydedir. Kısacası Lag2 anlamlıdır, Lag1 ise sınıra yakın ama anlamlı değildir.

2.3 Weekly için logistic regresyon modelinin kurulumu ve sonuçlar

```r
# Modelden olasılık tahmini yapıyoruz
prob <- predict(model, type = "response")

# Olasılığa göre tahmini sınıfları belirliyoruz (eşik: 0.5)
predicted_direction <- ifelse(prob > 0.5, "Up", "Down")
predicted_direction <- factor(predicted_direction, levels = c("Down", "Up"))

# Gerçek sınıflar
actual_direction <- Weekly$Direction

# Conf matrisi oluşturma
table(Predicted = predicted_direction, Actual = actual_direction)

conf_matrix <- table(Predicted = predicted_direction, Actual = actual_direction)
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
accuracy
```

Şekil 9: İlgili modelin R'da kurulumu

```
> table(Predicted = predicted_direction, Actual = actual_direction)
        Actual
Predicted Down  Up
     Down   54  48
     Up    430 557
> conf_matrix <- table(Predicted = predicted_direction, Actual = actual_direction)
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> accuracy
[1] 0.5610652
```

Şekil 10: İlgili modelin sonuçları

Lojistik regresyon modeli, eğitim verisinde %56.1 doğruluk sağladı ancak bu oran, sadece "her hafta yükselir" tahmini yaparak elde edilecek doğruluğa çok yakın. Model, yükseliş haftalarını oldukça iyi tahmin ederken (recall %92), düşüş haftalarını genellikle yanlış tahmin ediyor (false positive rate %88.8). Bu durum, yatırım gibi riskli alanlarda önemli bir sorun oluşturabilir. Ayrıca, precision ve NPV değerleri de ancak rastgele tahmin seviyesinde kalıyor.

Sadece Lag2 ile kurulan model:

```r
# Eğitim: 1990-2008
train <- subset(Weekly, Year < 2009)

# Test: 2009-2010
test <- subset(Weekly, Year >= 2009)

model_lag2 <- glm(Direction ~ Lag2, data = train, family = binomial)
summary(model_lag2)

# Test için olasılık tahmini
prob_test <- predict(model_lag2, newdata = test, type = "response")

# Tahmini sınıflar (eşik 0.5)
pred_test <- ifelse(prob_test > 0.5, "Up", "Down")
pred_test <- factor(pred_test, levels = c("Down", "Up"))

# Gerçek test sınıfları
actual_test <- test$Direction

# Karışıklık matrisi
conf_matrix_test <- table(Predicted = pred_test, Actual = actual_test)
print(conf_matrix_test)

# Doğru sınıflandırma oranı
accuracy_test <- sum(diag(conf_matrix_test)) / sum(conf_matrix_test)
accuracy_test
```

Şekil 11:Lag2 ile kurulan modelin kod satırı

```
Call:
glm(formula = Direction ~ Lag2, family = binomial, data = train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.536  -1.264   1.021   1.091   1.368

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.20326    0.06428   3.162  0.00157 **
Lag2         0.05810    0.02870   2.024  0.04298 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1350.5  on 983  degrees of freedom
AIC: 1354.5

> print(conf_matrix_test)
         Actual
Predicted Down Up
     Down    9  5
     Up     34 56
> # Doğru sınıflandırma oranı
> accuracy_test <- sum(diag(conf_matrix_test)) / sum(conf_matrix_test)
> accuracy_test
[1] 0.625
```

Şekil 12 ve 13 :Sonuç çıktıları

2008'e kadar olan verilerle yalnızca **Lag2** değişkeni kullanılarak kurulan lojistik regresyon modeli, 2009–2010 verilerinde piyasa yönünü %62.5 doğrulukla tahmin etmiştir. Bu oran şansa göre daha iyi olsa da, her haftayı "Up" olarak tahmin eden basit bir modelden yalnızca %10 kadar daha başarılıdır.

"Up" haftaları pozitif sınıf olarak ele alırsak:

- **Doğru pozitif oranı (TPR)**: 56/61 ≈ **%91.8**
- **Yanlış pozitif oranı (FPR)**: 34/43 ≈ **%79.1**
- **Pozitif kestirim değeri (Precision)**: 56/90 ≈ **%62.2**
- **Negatif kestirim değeri (NPV)**: 9/14 ≈ **%64.3**

Sonuç olarak, sadece Lag2 ile kurulan model az da olsa başarılı, ancak yanlış pozitif oranı yüksek olduğu için yatırım kararlarında değerlendirilmelidir.

2.4 Weekly için diğer modellerle yapılan denemeler

Bu kısımda, sınıflandırma problemlerinde üç farklı makine öğrenmesi algoritması olan Lojistik Regresyon, K-En Yakın Komşu (K=1) ve Naive Bayes yöntemleri karşılaştırılmıştır. Modeller eğitim verisiyle eğitilip test verisi üzerinde değerlendirilmiş, her biri için karışıklık matrisi ve doğruluk oranı hesaplanmıştır. Amaç, bu temel sınıflandırma algoritmalarının doğruluk performanslarını gözlemleyerek veri seti üzerinde hangisinin daha etkili sonuçlar verdiğini analiz etmektir. Bu işlemler için python kullanılmıştır.

```python
# --- a) Lojistik Regresyon ---
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)

cm_logreg = confusion_matrix(y_test, y_pred_logreg)
acc_logreg = accuracy_score(y_test, y_pred_logreg)
print("Lojistik Regresyon Karışıklık Matrisi:\n", cm_lo
print("Lojistik Regresyon Doğruluk Oranı:", acc_logreg)

# --- d) K-NN (K=1) ---
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

cm_knn = confusion_matrix(y_test, y_pred_knn)
acc_knn = accuracy_score(y_test, y_pred_knn)
print("\nK-NN (K=1) Karışıklık Matrisi:\n", cm_knn)
print("K-NN Doğruluk Oranı:", acc_knn)

# --- f) Naive Bayes ---
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)

cm_nb = confusion_matrix(y_test, y_pred_nb)
acc_nb = accuracy_score(y_test, y_pred_nb)
print("\nNaive Bayes Karışıklık Matrisi:\n", cm_nb)
print("Naive Bayes Doğruluk Oranı:", acc_nb)
```

Şekil 14: Diğer modellerin pythonda kurulumu

```
Lojistik Regresyon Karışıklık Matrisi:
 [[ 9 34]
 [ 5 56]]
Lojistik Regresyon Doğruluk Oranı: 0.625

K-NN (K=1) Karışıklık Matrisi:
 [[22 21]
 [30 31]]
K-NN Doğruluk Oranı: 0.5096153846153846

Naive Bayes Karışıklık Matrisi:
 [[ 0 43]
 [ 0 61]]
Naive Bayes Doğruluk Oranı: 0.5865384615384616
```

Şekil 15: İlgili çıktılar

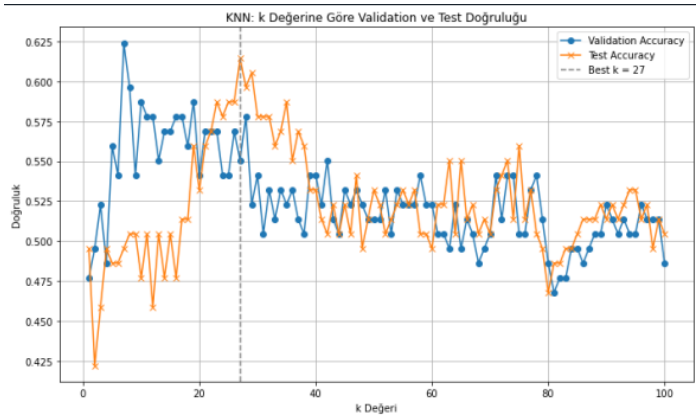
Sadece lag2 ile kurulan modellerde sonuçlar Şekil 15'teki gibi gelmektedir. Bundan farklı olarak sadece lag2 parametresi yerine Lag1 %40, Lag2 %35, Lag3 %15, Lag4 ve Lag5 ise %5'er şeklinde sezgiler olarak farklı bir modelde ve 2008 sonrası test olarak ayırmak yerine genel tüm dataları kapsayacak şekilde train–valid ve test kümesi oluşturuldu ve bu datalar normalize edildi. Ayrıca farklı n sayısı denemeleri yapıldı.

```
weighted.lag.avg = 0.4*Weekly$Lag1 +
0.35*Weekly$Lag2 + 0.15*Weekly$Lag3 +
0.05*Weekly$Lag4 + 0.05*Weekly$Lag5
```

Şekil 16: Farklı lag yaklaşımı

```
# Train (80%), Validation (10%), Test (10%) bölme
X_train_full, X_temp, y_train_full, y_temp = train_test_split(X, y, test_size=0.2
X_valid, X_test, y_valid, y_test = train_test_split(X_temp, y_temp, test_size=0.5

# Normalizasyon
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_full)
X_valid_scaled = scaler.transform(X_valid)
X_test_scaled = scaler.transform(X_test)
```

Şekil 17: Normalizasyon ve kümeleri bölme



Şekil 18: K-değerlerine göre acc grafiği

```
En iyi k (test doğruluğuna göre): 27
Validation doğruluğu (best_k için): 0.550
Test doğruluğu: 0.615
```

Şekil 19: K-değerlerine en iyi acc sonuçları

2.4.1 Sonuç ve Değerlendirme:

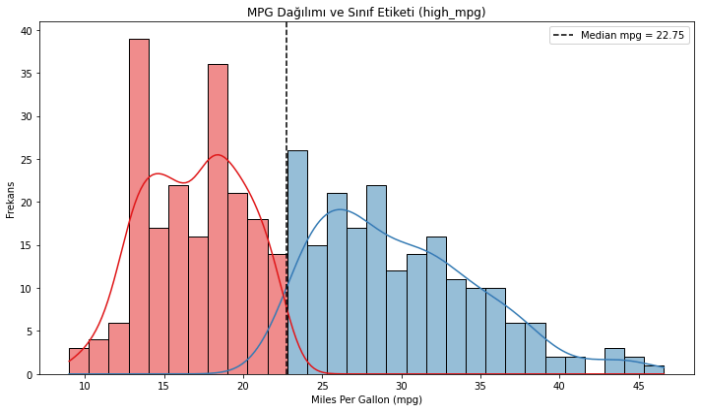
Yapılan üç yöntem karşılaştırıldığında, lojistik regresyon %62.5 doğrulukla en iyi performansı göstermiştir. K-En Yakın Komşu (K=1) yöntemi %51 civarında doğrulukla en düşük başarıyı göstermiştir. Naive Bayes yöntemi ise %58 civarında orta düzeyde bir doğruluk sağlamıştır. Diğer denemeler sonucunda en iyi K değeri 27 bulunmuş ve bu değerle KNN doğruluğu test verisinde %61.5'e çıkarken lojistik regresyonun ve naive bayesin doğruluk oranı azalmıştır. Up ve down sınıf dengesizliği yüzünden bu oran en fazla bu kadar artıyor olabilir bunun dışında dataya farklı yaklaşımlar uygulanarak bu oranlar arttırılabilir.

3 Auto data seti ve görselleştirme

Bu kısımda, Auto veri seti kullanılarak araçların yakıt verimliliğinin yüksek mi düşük mü olduğunu tahmin etmek amacıyla destek vektör makineleri (SVM) uygulanmıştır. İlk olarak, mpg (yakıt verimliliği) değişkeninin medyan değeri baz alınarak, bu değerin üzerinde olan araçlar 1, altında olanlar ise 0 olarak ikili hedef değişken oluşturulmuştur. Daha sonra, mpg değişkeni hariç tüm sayısal özellikler bağımsız değişken olarak kullanılarak, farklı C parametre değerleriyle lineer SVM modelleri 5 katlı çapraz doğrulama ile değerlendirilmiştir. Ardından, radial (RBF) ve polinom çekirdek fonksiyonları kullanılarak, gamma, derece ve C parametreleri çeşitli değerlerde denenmiş; bu modellerin çapraz doğrulama doğrulukları hesaplanarak karşılaştırılmıştır. Son olarak, parametre değişimlerinin model performansına etkisini görselleştirmek amacıyla uygun grafikler oluşturulmuş, böylece farklı parametre ayarlarının doğruluk üzerindeki etkileri detaylı şekilde analiz edilmiştir. Görselleştirmelerde, açıklayıcı değişken sayısının fazlalığı nedeniyle ikili değişken kombinasyonları kullanılarak SVM karar sınırları incelenmiştir.

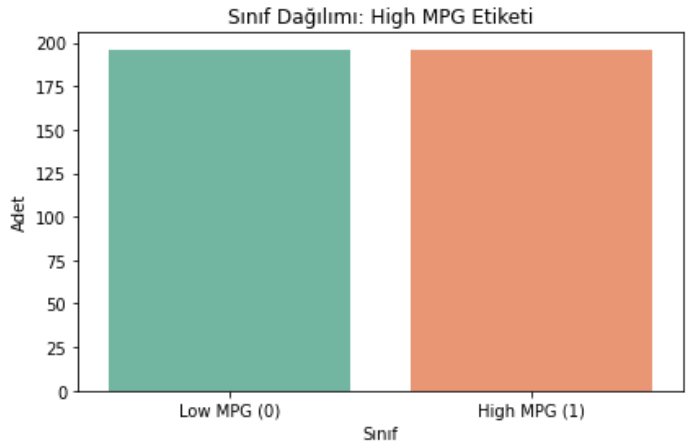
index	mpg	cylinders	isplacemer	orsepowe	weight	cceleration	year	origin	name	high_mpg
194	24.5	4	98	60	2164	22.1	76	1	chevrolet woody	1
195	29	4	90	70	1937	14.2	76	2	vw rabbit	1
196	33	4	91	53	1795	17.4	76	3	honda civic	1
197	20	6	225	100	3651	17.7	76	1	dodge aspen se	0
198	18	6	250	78	3574	21	76	1	ford granada ghia	0
199	18.5	6	250	110	3645	16.2	76	1	pontiac ventura sj	0
200	17.5	6	258	95	3193	17.8	76	1	amc pacer d/l	0
201	29.5	4	97	71	1825	12.2	76	2	volkswagen rabbit	1
202	32	4	85	70	1990	17	76	3	datsun b-210	1
203	28	4	97	75	2155	16.4	76	3	toyota corolla	1
204	26.5	4	140	72	2565	13.6	76	1	ford pinto	1

Şekil 20: Yapılan işlem sonrası oluşan dataframe



Şekil 21: Mpg değerlerinin dağılımı

Şekil 20 ve 21, Auto.csv veri kümesindeki araçların yakıt verimliliğini temsil eden mpg (miles per gallon) değişkeni kullanılarak basit bir ikili sınıflandırma yapılmıştır. Verideki eksik değerler temizlendikten sonra, mpg değerlerinin medyanı 22.75 olarak hesaplanmış ve bu değer eşik alınarak araçlar high_mpg (1: verimli) ve low_mpg (0: verimsiz) olarak iki sınıfa ayrılmıştır. Oluşturulan histogram, mpg dağılışını ve bu sınıfları görsel olarak sunarken, medyan değer kesikli siyah çizgi ile gösterilmiştir. Grafik, düşük mpg değerlerinin daha yoğun olduğunu ve dağılımın sağa çarpık bir yapı sergilediğini ortaya koymakta, böylece araçların çoğunluğunun ortalama ve ortalamanın biraz altında yakıt verimliliğine sahip olduğunu göstermektedir. Bu tür bir etiketleme, sınıflandırma algoritmaları için temel bir hedef değişken oluşturmak amacıyla kullanılır.



Şekil 22: 0 ve 1 sınıflarının dağılımı

```python
# Veriyi indir
url = "https://raw.githubusercontent.com/selva86/datasets/master/Auto.csv"
auto = pd.read_csv(url, na_values='?').dropna()

# Medyan mpg hesapla
median_mpg = auto['mpg'].median()
print(f"Medyan MPG: {median_mpg:.2f}")

# Binary sınıf oluştur
auto['high_mpg'] = (auto['mpg'] > median_mpg).astype(int)

# Genel görselleştirme
plt.figure(figsize=(10, 6))
sns.histplot(data=auto, x='mpg', hue='high_mpg', bins=30, kde=True, palette='Set1')
plt.axvline(median_mpg, color='black', linestyle='--', label=f'Median mpg = {median_mpg:.2f}')
plt.title('MPG Dağılımı ve Sınıf Etiketi (high_mpg)')
plt.xlabel('Miles Per Gallon (mpg)')
plt.ylabel('Frekans')
plt.legend()
plt.tight_layout()
plt.show()

# 0 ve 1 ler
plt.figure(figsize=(6, 4))
sns.countplot(x='high_mpg', data=auto, palette='Set2')
plt.xticks([0, 1], ['Low MPG (0)', 'High MPG (1)'])
plt.title('Sınıf Dağılımı: High MPG Etiketi')
plt.xlabel('Sınıf')
plt.ylabel('Adet')
plt.tight_layout()
plt.show()
```

Şekil 23: İlgili kod satırı

D. 3.1 Svm ile modelleme

SVM sınıflandırması için üç farklı çekirdek (kernel) fonksiyonu test edilmiştir: **doğrusal (linear)**, **RBF (radial basis function)** ve **polinomsal (polynomial)**. Model oluştururken oluşturduğunuz yeni değişkeni, bağımsız değişkenler olarak ise mpg dışındaki tüm sayısal değişkenleri kullanıyoruz.

- **Linear kernel** için sadece C parametresi değiştirilmiş ve C = [0.01, 0.1, 1, 10, 100] değerleri denenmiştir. Bu parametre, modelin hata toleransını kontrol ederken doğruluk değerleri incelenmiştir.
- **RBF kernel** ile hem C hem de gamma parametreleri kombine şekilde test edilmiştir. C = [0.01, 0.1, 1, 10, 100] ve gamma = [0.001, 0.01, 0.1, 1, 10] olmak üzere toplam 25 farklı kombinasyon denenmiştir. Gamma, örneklerin etki alanını belirleyerek karar sınırlarının karmaşıklığını etkiler.
- **Polynomial kernel** için C değerleri yine aynı aralıkta sabit tutularak, bu sefer degree = [2, 3, 4, 5] değerleri denenmiştir. Degree, polinom fonksiyonunun derecesini belirler ve modelin karar sınırlarının doğrusal olmaktan ne kadar uzaklaşacağını gösterir.

Her kombinasyon için model 5 katlı çapraz doğrulama (5-fold cross-validation) ile değerlendirilmiş ve ortalama doğruluk skorları hesaplanarak çıktılanmıştır. Bu kapsamlı hiperparametre taraması, farklı kernel tipleri altında hangi ayarların sınıflandırma performansını en iyi şekilde optimize ettiğini karşılaştırmalı olarak analiz etmeye olanak tanır.

```python
X = auto.select_dtypes(include=[np.number]).drop(columns=['mpg'])
y = auto['high_mpg']

C_values = [0.01, 0.1, 1, 10,100]
gamma_values = [0.001, 0.01, 0.1, 1, 10]
degree_values = [2, 3, 4, 5]

print("Linear Kernel:")
for c in C_values:
    svm = SVC(C=c, kernel='linear')
    scores = cross_val_score(svm, X, y, cv=5)
    print(f"C = {c}, Ortalama Doğruluk: {scores.mean():.4f}")

print("RBF Kernel için C-Gamma kombinasyonlarının doğrulukları:\n")
for c in C_values:
    for g in gamma_values:
        svm = SVC(C=c, kernel='rbf', gamma=g)
        scores = cross_val_score(svm, X, y, cv=5)
        print(f"C = {c:<5} Gamma = {g:<5} => Ortalama Doğruluk: {scores.mean():.4f}")

print("Polynomial Kernel için C-Degree kombinasyonlarının doğrulukları:\n")
for c in C_values:
    for d in degree_values:
        svm = SVC(C=c, kernel='poly', degree=d)
        scores = cross_val_score(svm, X, y, cv=5)
        print(f"C = {c:<5} Degree = {d:<2} => Ortalama Doğruluk: {scores.mean():.4f}")
```

Şekil 24: SVM modeli oluşturma

Linear Kernel için C ile doğruluklar:
C = 0.01, Ortalama Doğruluk: 0.8853
C = 0.1, Ortalama Doğruluk: 0.9974
C = 1, Ortalama Doğruluk: 1.0000
C = 10, Ortalama Doğruluk: 1.0000
C = 100, Ortalama Doğruluk: 1.0000

RBF Kernel için C-Gamma kombinasyonlarının doğrulukları:
C = 0.01 Gamma = 0.001 => Ortalama Doğruluk: 0.6257
C = 0.01 Gamma = 0.01 => Ortalama Doğruluk: 0.5872
C = 0.01 Gamma = 0.1 => Ortalama Doğruluk: 0.5436
C = 0.01 Gamma = 1 => Ortalama Doğruluk: 0.5231
C = 0.01 Gamma = 10 => Ortalama Doğruluk: 0.5052
C = 0.1 Gamma = 0.001 => Ortalama Doğruluk: 0.7193
C = 0.1 Gamma = 0.01 => Ortalama Doğruluk: 0.5872
C = 0.1 Gamma = 0.1 => Ortalama Doğruluk: 0.5436
C = 0.1 Gamma = 1 => Ortalama Doğruluk: 0.5231
C = 0.1 Gamma = 10 => Ortalama Doğruluk: 0.5052
C = 1 Gamma = 0.001 => Ortalama Doğruluk: 0.9029
C = 1 Gamma = 0.01 => Ortalama Doğruluk: 0.7346
C = 1 Gamma = 0.1 => Ortalama Doğruluk: 0.5358
C = 1 Gamma = 1 => Ortalama Doğruluk: 0.5000
C = 1 Gamma = 10 => Ortalama Doğruluk: 0.4975
C = 10 Gamma = 0.001 => Ortalama Doğruluk: 0.8724
C = 10 Gamma = 0.01 => Ortalama Doğruluk: 0.7397
C = 10 Gamma = 0.1 => Ortalama Doğruluk: 0.5358
C = 10 Gamma = 1 => Ortalama Doğruluk: 0.5000
C = 10 Gamma = 10 => Ortalama Doğruluk: 0.4975
C = 100 Gamma = 0.001 => Ortalama Doğruluk: 0.8647
C = 100 Gamma = 0.01 => Ortalama Doğruluk: 0.7397
C = 100 Gamma = 0.1 => Ortalama Doğruluk: 0.5358
C = 100 Gamma = 1 => Ortalama Doğruluk: 0.5000
C = 100 Gamma = 10 => Ortalama Doğruluk: 0.4975

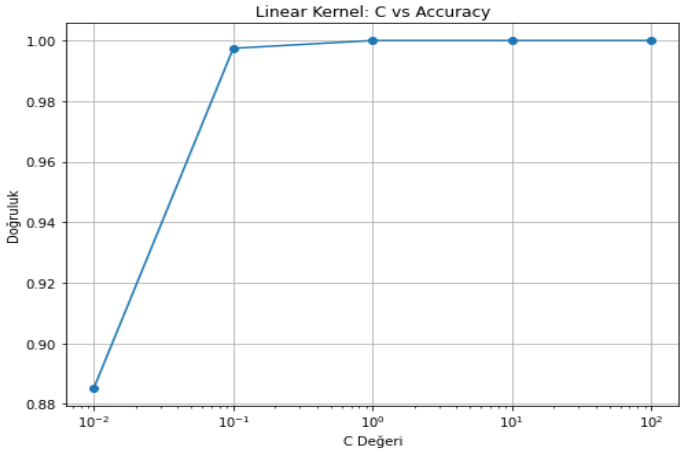
Polynomial Kernel için C-Degree kombinasyonlarının doğrulukları:

C = 0.01 Degree = 2 => Ortalama Doğruluk: 0.7931
C = 0.01 Degree = 3 => Ortalama Doğruluk: 0.8110
C = 0.01 Degree = 4 => Ortalama Doğruluk: 0.8314
C = 0.01 Degree = 5 => Ortalama Doğruluk: 0.8391
C = 0.1 Degree = 2 => Ortalama Doğruluk: 0.8749
C = 0.1 Degree = 3 => Ortalama Doğruluk: 0.8774
C = 0.1 Degree = 4 => Ortalama Doğruluk: 0.8723
C = 0.1 Degree = 5 => Ortalama Doğruluk: 0.8723
C = 1 Degree = 2 => Ortalama Doğruluk: 0.8748
C = 1 Degree = 3 => Ortalama Doğruluk: 0.8723
C = 1 Degree = 4 => Ortalama Doğruluk: 0.8799
C = 1 Degree = 5 => Ortalama Doğruluk: 0.8775
C = 10 Degree = 2 => Ortalama Doğruluk: 0.8800
C = 10 Degree = 3 => Ortalama Doğruluk: 0.8851
C = 10 Degree = 4 => Ortalama Doğruluk: 0.8851
C = 10 Degree = 5 => Ortalama Doğruluk: 0.8825
C = 100 Degree = 2 => Ortalama Doğruluk: 0.8902
C = 100 Degree = 3 => Ortalama Doğruluk: 0.8979
C = 100 Degree = 4 => Ortalama Doğruluk: 0.8954
C = 100 Degree = 5 => Ortalama Doğruluk: 0.8953

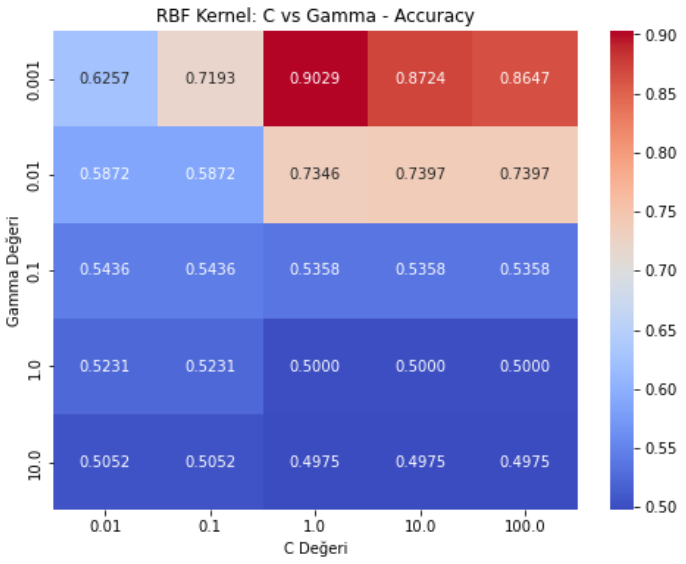
Elde edilen doğruluk sonuçları, farklı çekirdek fonksiyonları (kernel) ve hiperparametre ayarlarının model başarımı üzerinde ne denli etkili olduğunu açıkça göstermektedir. Ayrıca yorumlamızın daha kolay olması adına sonuçları aşağıda görselleştireceğiz. Fakat poly ve rbf kerneli için 3 ayrı parametre olduğu için en kolay görselleştirme metodu olarak heatmap şeklinde yapmamız daha anlamlı olacak.

```
# Linear Kernel Grafik
plt.figure(figsize=(8,6))
plt.plot(linear_df['C'], linear_df['Accuracy'], marker='o')
plt.title("Linear Kernel: C vs Accuracy")
plt.xscale('log')
plt.xlabel("C Değeri")
plt.ylabel("Doğruluk")
plt.grid(True)
plt.show()

# RBF Kernel Heatmap
plt.figure(figsize=(8,6))
rbf_pivot = rbf_df.pivot(index='Gamma', columns='C', values='Accuracy')
sns.heatmap(rbf_pivot, annot=True, fmt=".4f", cmap='coolwarm')
plt.title("RBF Kernel: C vs Gamma - Accuracy")
plt.xlabel("C Değeri")
plt.ylabel("Gamma Değeri")
plt.show()

# Polynomial Kernel Heatmap
plt.figure(figsize=(8,6))
poly_pivot = poly_df.pivot(index='Degree', columns='C', values='Accuracy')
sns.heatmap(poly_pivot, annot=True, fmt=".4f", cmap='viridis')
plt.title("Polynomial Kernel: C vs Degree - Accuracy")
plt.xlabel("C Değeri")
plt.ylabel("Degree")
plt.show()
```

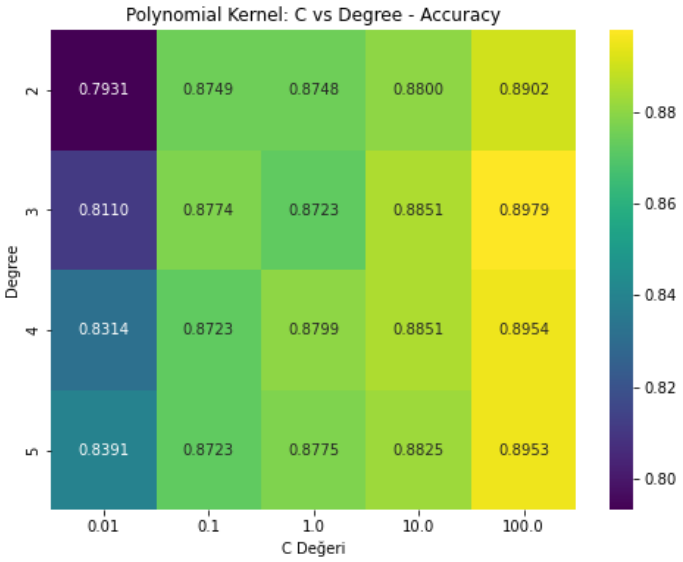
Şekil 25:İlgili kod satırı



Şekil 26:Linear Kernel: C ve Accuracy ilişkisi



Şekil 27:RBF Kernel: C vs Gamma - Accuracy ilişkisi



Şekil 27:Polynomial Kernel: C vs Gamma - Accuracy ilişkisi

Aşağıda her bir kernel türüne özel detaylı yorum yer almaktadır:

Linear Kernel

- C değeri arttıkça doğruluk anlamlı biçimde yükselmiş;
    - C = 0.01 için doğruluk %88.5 iken,
    - C = 0.1 ile %99.7'ye,
    - C >= 1 için ise doğruluk %100 olmuştur.
- Bu, verinin lineer olarak oldukça iyi ayrılabildiğini ve az ceza (C küçük) ile düşük performans alınırken, daha yüksek C değerleriyle mükemmel ayraç çizilebildiğini gösteriyor.
- Overfitting riski olsa da, doğrusal kernelin bu veri kümesinde çok uygun bir seçim olduğu anlaşılmaktadır.

RBF (Radial Basis Function) Kernel

- Düşük C ve yüksek gamma kombinasyonlarında modelin doğruluğu belirgin şekilde düşüktür. Örneğin:
    - C=0.01, gamma=10 için doğruluk sadece %50 civarındadır (şansa yakın).
- En iyi performanslar:
    - C=1, gamma=0.001 için %90.3,
    - C=10, gamma=0.001 için %87.2 doğruluk elde edilmiştir.
- Bu sonuçlar, RBF kernelin yalnızca **uygun C ve gamma değerleri** ile başarılı sonuçlar verdiğini ve ayarlara çok duyarlı olduğunu göstermektedir.
- Genel olarak linear kernel kadar başarılı değildir, özellikle gamma değeri çok büyük olduğunda aşırı öğrenme (overfitting) nedeniyle doğruluk hızla düşmektedir.

Polynomial Kernel

- Polinomsal kernelin doğruluğu, C ve degree parametreleri arttıkça **kademeli olarak iyileşmektedir**.
    - C=0.01 için degree arttıkça doğruluk %79'dan %84'e çıkar.
    - C=100, degree=3 için en yüksek doğruluk %89.8'e ulaşır.

- Bu kernel tipi, doğrusal kernel kadar mükemmel bir ayırım yapmasa da tutarlı ve yüksek doğruluklar sağlamaktadır.
- Özellikle C=10 ve C=100 ile degree=3-5 aralığında iyi sonuçlar alınmıştır, bu da veri yapısında **doğrusal olmayan bazı ilişkilerin** de olabileceğine işaret eder.

Genel Değerlendirme:

- **En yüksek doğruluk** linear kernel ile `C ≥ 1` olduğunda (%100) elde edilmiştir.
- **RBF kernel**, ayarlara çok duyarlı olup yalnızca `gamma` çok küçükken (`0.001`) makul doğruluklar vermektedir.
- **Polynomial kernel**, `C` ve `degree` arttıkça performansı artmış ve %89.8'e kadar çıkmıştır; ancak linear kernel ile kıyaslandığında daha fazla ayarlama gerektirmiştir.

3.2 Yorumlama:

Bu veri seti için **en uygun model** yüksek `C` değerine sahip **linear kernel**'li SVM'dir. Model veriyi neredeyse kusursuz ayırabilmektedir. Bu da sınıflar arasında lineer ayırma olanağının oldukça güçlü olduğuna ve karmaşık kernel yapılarına gerek kalmadan yüksek başarı elde edilebileceğine işaret eder. RBF ve polynomial kernel'ler daha esnek olmakla birlikte, bu özel durumda lineer çözüm en etkin yaklaşımdır.

III. *SONUÇ*

Bu çalışmada, finansal ve otomotiv alanlarında yaygın olarak kullanılan Weekly ve Auto veri setleri üzerinde temel makine öğrenmesi sınıflandırma algoritmalarının performansları kapsamlı şekilde incelenmiştir. Weekly veri setinde lojistik regresyon, K-En Yakın Komşu (KNN) ve Naive Bayes gibi yöntemler denenmiş, Auto veri setinde ise farklı çekirdek fonksiyonlarına sahip destek vektör makineleri (SVM) kullanılarak sınıflandırma başarısı değerlendirilmiştir.

Analizler göstermiştir ki, Weekly veri setinde Lag2 değişkeni istatistiksel olarak anlamlı bulunmuş ve sadece bu değişkenle kurulan lojistik regresyon modeli, test verisinde (%62.5) doğruluk ile diğer yöntemlere kıyasla en iyi performansı göstermiştir. KNN ve Naive Bayes yöntemleri ise, veri setinin sınıf dengesizliği ve finansal zaman serilerinin doğası gereği, nispeten daha düşük doğruluk oranlarına ulaşmıştır. Ayrıca, farklı lag kombinasyonları ve parametre ayarları ile yapılan denemelerde, veri setinin temel yapısındaki sınırlamalar nedeniyle doğruluk oranlarının sınırlandığı gözlenmiştir.

Auto veri setinde gerçekleştirilen SVM modellemelerinde ise, lineer kernel ile C parametresi artırıldıkça doğruluk oranı %100'e kadar ulaşmış ve bu, veri setinin lineer olarak ayrılabilir olduğunu göstermiştir. RBF ve polinomsal kernel uygulamalarında ise doğruluk oranları, C, gamma ve degree parametrelerine göre değişkenlik göstermiştir. Özellikle RBF kernel, sadece uygun hiperparametre kombinasyonlarında yüksek doğruluk sağlamış, aksi halde model doğruluğu anlamlı şekilde düşmüştür. Polinomsal kernel ise, parametreler arttıkça istikrarlı bir şekilde daha yüksek doğruluklara ulaşmıştır.

Genel olarak, bu çalışma farklı alanlardan iki veri seti üzerinde temel sınıflandırma algoritmalarının uygulanabilirliğini ve başarılarını karşılaştırmalı şekilde ortaya koymuştur. Sonuçlar, model ve hiperparametre seçiminin veri setinin yapısına ve problem türüne göre dikkatlice yapılması gerektiğini göstermektedir. Ayrıca, sınıf dengesizliği, değişken seçimi ve parametrik ayarların model başarısı üzerinde önemli etkileri olduğu bir kez daha vurgulanmıştır. Elde edilen bulgular, gerçek dünya veri setleri üzerinde makine öğrenmesi yöntemlerinin uygulanmasında yol gösterici niteliktedir.