knn-dt-svm

Berat Mert Kayacan

Dataset Generation –> student_id=2365718

```r
library(class)       # KNN
library(rpart)       # Decision Tree
library(rpart.plot)  # Decision Tree Visualization
library(e1071)       # SVM


generate_data <- function(student_id) {
  set.seed(student_id)
  n <- 1000

  age                <- round(runif(n, 18, 65))
  income             <- pmax(rnorm(n, mean=50000, sd=15000), 10000)
  years_of_education <- round(runif(n, 8, 22))
  prev_purchases     <- pmin(rpois(n, lambda=3), 10)
  discount           <- sample(c('yes','no'), n, replace=TRUE, prob=c(0.4, 0.6))
  free_delivery      <- sample(c('yes','no'), n, replace=TRUE, prob=c(0.3, 0.7))
  product_price      <- round(runif(n, 10, 500), 2)
  product_rating     <- round(runif(n, 1, 5), 1)

  income_scaled <- (income - 10000) / 90000
  prev_scaled   <- prev_purchases / 10
  edu_scaled    <- (years_of_education - 8) / 14
  price_norm    <- (product_price - 10) / 490
  rating_scaled <- (product_rating - 1) / 4
  disc_bin      <- ifelse(discount=='yes', 1, 0)
  fd_bin        <- ifelse(free_delivery=='yes', 1, 0)

  logit <- -0.5 + 1.5*income_scaled + 1.2*prev_scaled +
           0.8*disc_bin + 0.6*fd_bin - 1.0*price_norm +
           0.7*rating_scaled + 0.4*edu_scaled +
           rnorm(n, 0, 0.3)
  prob     <- 1 / (1 + exp(-logit))
  purchase <- ifelse(runif(n) < prob, 'yes', 'no')

  data.frame(Age=age, Income=round(income,2),
             YearsOfEducation=years_of_education,
             PreviousPurchases=prev_purchases,
             DiscountOffered=discount, FreeDelivery=free_delivery,
             ProductPrice=product_price, ProductRating=product_rating,
             Purchase=purchase)
```

```
}
data <- generate_data(student_id = 2365718)
head(data)
```

```
##   Age   Income YearsOfEducation PreviousPurchases DiscountOffered FreeDelivery
## 1  45 45914.09               22                 6             yes           no
## 2  54 26086.64               10                 2              no           no
## 3  62 53735.44                9                 2              no           no
## 4  47 52439.13               21                 5              no           no
## 5  57 36517.86                9                 4              no          yes
## 6  56 32096.85               15                 4              no           no
##   ProductPrice ProductRating Purchase
## 1        26.03           3.6      yes
## 2       291.81           4.1      yes
## 3        50.51           4.1      yes
## 4       379.68           4.4      yes
## 5       106.11           3.2      yes
## 6        14.26           4.2      yes
```

0. Preprocessing

```
# DiscountOffered, FreeDelivery, Purchase değişkenleri encoding (1/0'a çevir)
data$DiscountOffered <- ifelse(data$DiscountOffered == "yes", 1, 0)
data$FreeDelivery    <- ifelse(data$FreeDelivery == "yes", 1, 0)
data$Purchase <- as.factor(data$Purchase)   # 0=not purchase 1=purchase
str(data)
```

```
## 'data.frame':    1000 obs. of  9 variables:
##  $ Age              : num  45 54 62 47 57 56 22 41 24 39 ...
##  $ Income           : num  45914 26087 53735 52439 36518 ...
##  $ YearsOfEducation : num  22 10 9 21 9 15 15 11 14 21 ...
##  $ PreviousPurchases: num  6 2 2 5 4 4 2 4 3 3 ...
##  $ DiscountOffered  : num  1 0 0 0 0 0 0 0 1 0 ...
##  $ FreeDelivery     : num  0 0 0 0 1 0 0 1 1 1 ...
##  $ ProductPrice     : num  26 291.8 50.5 379.7 106.1 ...
##  $ ProductRating    : num  3.6 4.1 4.1 4.4 3.2 4.2 3.7 2.7 4 1.5 ...
##  $ Purchase         : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
```

```
# Training-test %80-%20 böl
set.seed(2365718)
train_index <- sample(1:nrow(data), 0.8 * nrow(data))
train_data  <- data[train_index, ]
test_data   <- data[-train_index, ]

# Sınıf dağılımı tablo
tablo <- table(data$Purchase)
cat("\nPurchase sayıları:\n")
```

```
##
## Purchase sayıları:
```

```r
cat("0 (Satın almadı)   :", tablo[1], "\n")
```

```
## 0 (Satın almadı)   : 271
```

```r
cat("1 (Satın aldı)     :", tablo[2], "\n")
```

```
## 1 (Satın aldı)     : 729
```

```r
cat("Purchase Yes Oranı :", round(tablo[2]/sum(tablo)*100, 1),"%")
```

```
## Purchase Yes Oranı : 72.9 %
```

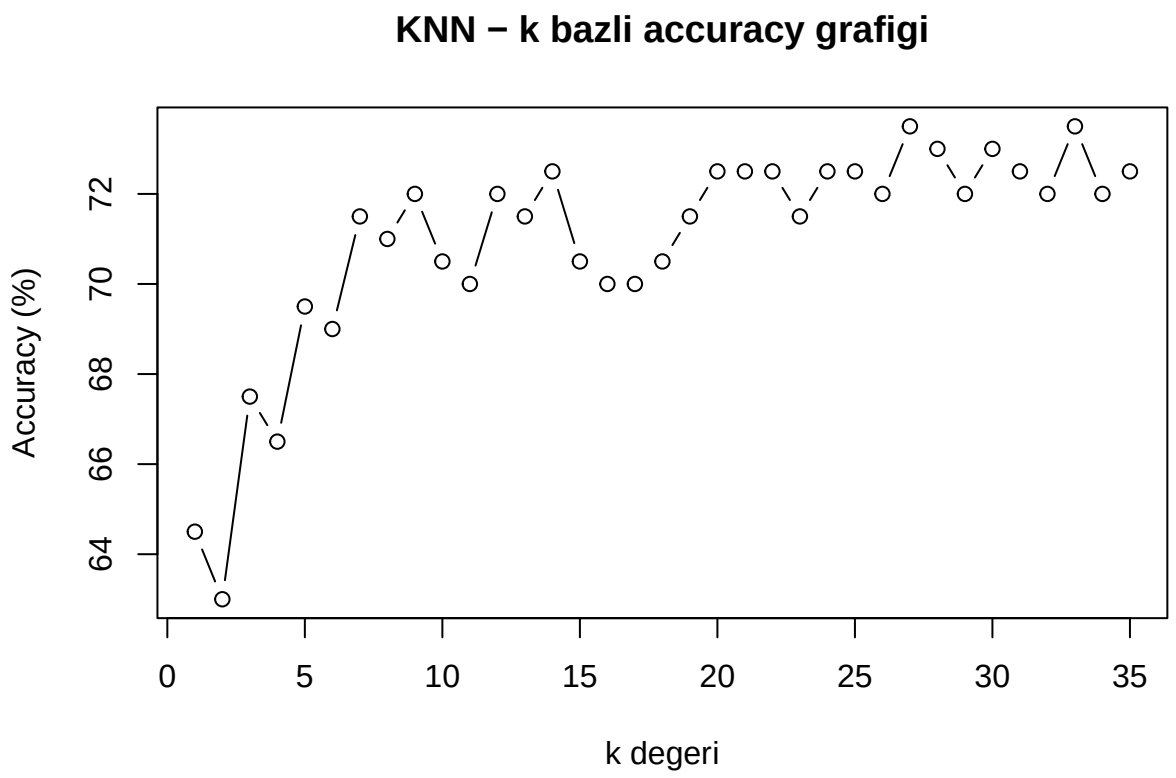
1. K-Nearest Neighbors (KNN)

```r
# feature: X ve label: Y
egitim_X <- train_data[, 1:8]
test_X   <- test_data[, 1:8]
egitim_Y <- train_data$Purchase
test_Y   <- test_data$Purchase

# Normalization (sadece eğitimden öğren)
ort <- apply(egitim_X, 2, mean)
std <- apply(egitim_X, 2, sd)
egitim_X <- scale(egitim_X, center = ort, scale = std)
test_X   <- scale(test_X,   center = ort, scale = std)

# En iyi k değeri - accuracy'e göre
accuracy <- numeric(35)
for (i in 1:35) {
  tahmin <- knn(egitim_X, test_X, egitim_Y, k = i)
  accuracy[i] <- mean(tahmin == test_Y) * 100
  cat("k =", i, " accuracy yüzdesi:", round(accuracy[i], 2), "%\n")
}
```

```
## k = 1  accuracy yüzdesi: 64.5 %
## k = 2  accuracy yüzdesi: 63 %
## k = 3  accuracy yüzdesi: 67.5 %
## k = 4  accuracy yüzdesi: 66.5 %
## k = 5  accuracy yüzdesi: 69.5 %
## k = 6  accuracy yüzdesi: 69 %
## k = 7  accuracy yüzdesi: 71.5 %
## k = 8  accuracy yüzdesi: 71 %
## k = 9  accuracy yüzdesi: 72 %
## k = 10  accuracy yüzdesi: 70.5 %
## k = 11  accuracy yüzdesi: 70 %
## k = 12  accuracy yüzdesi: 72 %
## k = 13  accuracy yüzdesi: 71.5 %
## k = 14  accuracy yüzdesi: 72.5 %
## k = 15  accuracy yüzdesi: 70.5 %
```

```
## k = 16  accuracy yüzdesi: 70 %
## k = 17  accuracy yüzdesi: 70 %
## k = 18  accuracy yüzdesi: 70.5 %
## k = 19  accuracy yüzdesi: 71.5 %
## k = 20  accuracy yüzdesi: 72.5 %
## k = 21  accuracy yüzdesi: 72.5 %
## k = 22  accuracy yüzdesi: 72.5 %
## k = 23  accuracy yüzdesi: 71.5 %
## k = 24  accuracy yüzdesi: 72.5 %
## k = 25  accuracy yüzdesi: 72.5 %
## k = 26  accuracy yüzdesi: 72 %
## k = 27  accuracy yüzdesi: 73.5 %
## k = 28  accuracy yüzdesi: 73 %
## k = 29  accuracy yüzdesi: 72 %
## k = 30  accuracy yüzdesi: 73 %
## k = 31  accuracy yüzdesi: 72.5 %
## k = 32  accuracy yüzdesi: 72 %
## k = 33  accuracy yüzdesi: 73.5 %
## k = 34  accuracy yüzdesi: 72 %
## k = 35  accuracy yüzdesi: 72.5 %
```

```r
# Visualization (grafik ile k bulma)
plot(accuracy, type = "b",
     xlab = "k degeri",
     ylab = "Accuracy (%)",
     main = "KNN - k bazli accuracy grafigi")
```



```r
# max accuracy 73.5 % ---> k=27 seçtik
k_final   <- 27
knn_final <- knn(egitim_X, test_X, egitim_Y, k = k_final)
cat("Final accuracy değeri:", mean(knn_final == test_Y),"\nFinal k = 27")
```

```
## Final accuracy değeri: 0.735
## Final k = 27
```

2. Decision Tree (DT)

```r
# 1,2,3,5,7,10 için maxdepth
derinlikler <- c(1, 2, 3, 5, 7, 10)
dt_acc      <- numeric(length(derinlikler))

for (j in 1:length(derinlikler)) {
  agac <- rpart(Purchase ~ ., data = train_data, method = "class",
                control = rpart.control(maxdepth = derinlikler[j], cp = 0))
  tahmin    <- predict(agac, test_data, type = "class")
  dt_acc[j] <- mean(tahmin == test_data$Purchase) * 100
  cat("maxdepth =", derinlikler[j], " accuracy:", round(dt_acc[j], 2), "%\n")
}
```

```
## maxdepth = 1  accuracy: 72 %
## maxdepth = 2  accuracy: 72 %
## maxdepth = 3  accuracy: 72 %
## maxdepth = 5  accuracy: 63 %
## maxdepth = 7  accuracy: 61 %
## maxdepth = 10  accuracy: 65 %
```

```r
# maxdepth = 3 ile final_agac
secilen_derinlik <- 3
final_agac <- rpart(Purchase ~ ., data = train_data, method = "class",
                    control = rpart.control(maxdepth = secilen_derinlik, cp = 0))

rpart.plot(final_agac, type = 2, fallen.leaves = TRUE) # rpart.plot ile görselleştirme
```


```r
# Final accuracy
final_tahmin <- predict(final_agac, test_data, type = "class")
cat(sprintf("Seçilen derinlik = %d | accuracy: %.2f%%\n", secilen_derinlik, mean(final_tahmin == test_da
```

```
## Seçilen derinlik = 3 | accuracy: 72.00%
```

3. Support Vector Machine (SVM)

```r
maliyetler <- c(0.01, 0.1, 1, 10, 100, 1000) # 6 değer için cost
svm_acc    <- numeric(length(maliyetler))

for (j in 1:length(maliyetler)) {
  model <- svm(Purchase ~ ., data = train_data,
               type   = "C-classification",
               kernel = "radial",
               cost   = maliyetler[j])
  tahmin     <- predict(model, test_data)
  svm_acc[j] <- mean(tahmin == test_data$Purchase) * 100
  cat("cost =", maliyetler[j], " accuracy:", round(svm_acc[j], 2), "%\n")
}
```

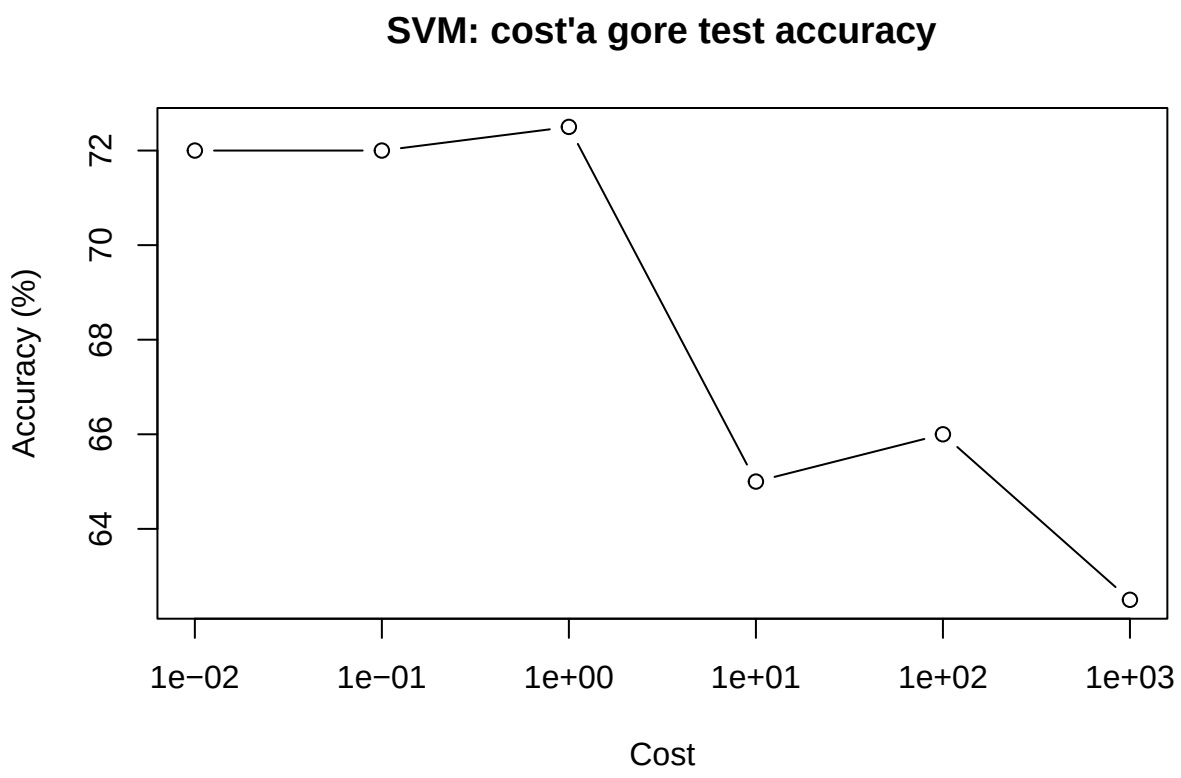
```
## cost = 0.01  accuracy: 72 %
```

```
## cost = 0.1  accuracy: 72 %
## cost = 1  accuracy: 72.5 %
## cost = 10  accuracy: 65 %
## cost = 100  accuracy: 66 %
## cost = 1000  accuracy: 62.5 %
```

```r
#DataFrame ile bakalım
sonuc_tablo <- data.frame(Cost = maliyetler, Accuracy = round(svm_acc, 2))
print(sonuc_tablo)
```

```
##    Cost Accuracy
## 1 1e-02     72.0
## 2 1e-01     72.0
## 3 1e+00     72.5
## 4 1e+01     65.0
## 5 1e+02     66.0
## 6 1e+03     62.5
```

```r
# Görselleştirme(cost geniş aralıkta log ile rahat)
plot(maliyetler, svm_acc, type = "b", log = "x",
     xlab = "Cost", ylab = "Accuracy (%)",
     main = "SVM: cost'a gore test accuracy")
```



```r
#final cost = 1 max accuracy(72.5%) bulundu
final_cost <- 1
svm_final  <- svm(Purchase ~ ., data = train_data,
                  type = "C-classification", kernel = "radial",
                  cost = final_cost)

svm_tahmin <- predict(svm_final, test_data)

table(Tahmin = svm_tahmin, Gercek = test_data$Purchase) #confusion matrix
```

```
##       Gercek
## Tahmin  no yes
##    no    5   4
##    yes  51 140
```

```r
cat(sprintf("\nFinal cost = %g | accuracy: %.2f%%\n", final_cost, mean(svm_tahmin == test_data$Purchase)
```

```
##
## Final cost = 1 | accuracy: 72.50%
```

Problem Tanımı

Analiz amacı müşteri ve ürün bilgilerini kullanarak bir müşterinin ürünü satın alıp almayacağını (Purchase = yes/no) tahmin etmektir. Binary yani yes no gibi ikili bir sınıflandırma problemidir. Veri seti öğrenci numaram 2365718 buna göre seed olarak olarak kullanılarak üretilmiş 1000 gözlem ve 8 feature'dan oluşmaktadır: Age, Income, YearsOfEducation, PreviousPurchases, DiscountOffered, FreeDelivery, ProductPrice ve ProductRating. Hedef değişkenin (Purchase-tahmin edilmek istenen değişkenin) sınıf dağılımı dataframe ile bakılınca dengesiz olduğu net: 729 yes (%72.9) ve 271 no (%27.1). Bu dengesizlik önemlidir çünkü trainden öğrenmeyip hepsine yes atayan bir model bile test setinde yaklaşık %72 doğruluk elde eder. Bu yüzden modelleri değerlendirirken bu %72'lik taban çizgisini referans almak önemliydi buna dikkat ettim.

Çözüm Metodolojisi

Preprocessing

Veri üretildikten sonra DiscountOffered ve FreeDelivery değişkenlerini 0/1 olacak şekilde sayısal değerlere çevirdim encoding yapılma sebebi KNN'de mesafe hesabı için featureların kategorik değil sayısal olması gerekli olduğundan. Hedef değişken Purchase'ı ise factor olarak tanımladım, çünkü Decision Tree ve SVM için hedefin factor olmasını ister. Datayı %80 training ve %20 test olacak şekilde böldüm ayrıca ise set.seed(2365718) ile tekrar üretilebilirlik sağlanmış oldu.

K-Nearest Neighbors (KNN)

KNN mesafe temelli bir yöntem olduğu için önce özelliklere standartization uyguladım (scale). Bunun nedeni, Income 50000.00 gibi bir sayı değeri alırken, ProductRating 1 ile 5 arası değerler alıyordu günün sonunda değişkenlerin ölçekleri çok farklıydı. Ölçeklemezsek Income tek başına mesafeyi belirlerdi. Ölçek parametrelerini yalnızca eğitim setinden öğrenip testi dışarıda bırakarak hem eğitime hem teste uygulamasını sağladım bunun sebebi de test verisinden bilgi sızmasını önlemek içindi.

Final k değerini seçmek için ise k = 1'den 35'e kadar tüm değerlerin accuracy skoruna bakarak denedim ve her biri için test doğruluğunu hesaplayıp grafikten de kontrol ettim ve gösterdim. Başlangıç referansı olarak sqrt(N) 31.6 kuralını kullandım. Grafiğe göre en yüksek accuracy skoru (%73.5) k = 27'de elde edildi. k = 27'yi seçtim çünkü: ilk olarak en yüksek doğruluğu veriyor, ikinci olarak tek sayı olduğu için binary classificatinda oylama berabere kalmamış oluyor. Çok küçük k (1–3) değerlerinden kaçındım çünkü bunlar outlier'lara aşırı duyarlıdır overfitting riski.

Decision Tree (DT)

Karar ağacı için rpart paketini kullandım ve ölçeklenmemiş orijinal veriyi verdim. Maxdepth değerini 1, 2, 3, 5, 7, 10 olarak değiştirip her biri için test accuracy değerini kontrol ettim ve şu şekilde çıktı:

maxdepth	Test Accuracy
1	72.0%
2	72.0%
3	72.0%
5	63.0%
7	61.0%
10	65.0%

Bunun sonucunda şu sonuca ulaştım: doğruluk(accuracy) derinlik 1–3 arasında en yüksek (%72) kalıyor, derinlik arttıkça düşüyor. Final derinlik olarak maxdepth = 3'ü seçtim: maksimum doğruluğun korunduğu en büyük derinlik olduğu için. Seçilen derinliğin ağacını rpart.plot ile görselleştirdim.

what it means when a tree is very deep. Is a deeper tree always better? Çok derin bir ağaç, eğitim verisindeki her detayı (outliers dahil) ezberler. Bu sebeple eğitimde yüksek doğruluk verse de test setinde veride başarısız olur yani overfitting olabilir. Sonuçlarımda da derinlik 3'ten sonra test doğruluğu düştü. Dolayısıyla daha derin ağaç her zaman daha iyi değildir. Belirli bir noktadan sonra model genelleme yeteneğini kaybeder.

Support Vector Machine (SVM)

SVM için e1071 paketini, C-classification türünü ve radial çekirdeği kullandım. svm() sayısal değişkenleri kendi içinde otomatik ölçeklediği için veriyi doğrudan verdim. En az 5 farklı cost değeri (0.01, 0.1, 1, 10, 100, 1000) denedim:

Cost	Test Accuracy
0.01	72.0%
0.1	72.0%
1	72.5%
10	65.0%
100	66.0%
1000	62.5%

Final cost değeri olarak 1'i seçtim çünkü en yüksek test doğruluğu (%72.5) bu cost değeri verdi ve orta bir değer olduğu için underfitting ile overfitting arasındaki dengeyi sağlamış oldu.

What happens to model performance when the cost is very high versus very low. Çok düşük cost (0.01–0.1) modeli fazla rahat yapar denebilir. Patterni yeterince öğrenemez yani underfitting ve doğruluk taban çizgisinde (%72) takılır. Çok yüksek cost (10–1000) ise modeli eğitim verisini ezberlemeye zorlar overfitting ve test doğruluğu düşer %62.5'a kadar düşmüş bizim durumumuzda. En iyi performans ikisinin arasındaki orta değer olan cost = 1'de elde edilir.

Comparison

Yöntem	Seçilen Parametre	Test Accuracy
KNN	k = 27	73.5%
Karar Ağacı	maxdepth = 3	72.0%
SVM	cost = 1	72.5%

Üç yöntem birbirine çok yakın sonuç verdi ve hepsi %72'lik taban çizgisinin yakınında kümelendi. En iyi performansı %73.5'luk accuracy değeri ile KNN gösterdi; taban çizgisini gözle görülür biçimde geçen tek yöntem KNN. SVM %72.5 ile ikinci, Decision Tree %72.0 ile taban çizgisiyle aynı sonucu verdi.

Bu sonuçların ana nedeni, hedef değişken üretilirken eklenen rastgele gürültü. Features ile Purchase arasındaki ilişkiyi büyük ölçüde maskeliyor. SVM'in karışıklık matrisinde model 56 gerçek "no" örneğinin yalnızca 5'ini doğru tahmin etti; yani çoğunlukla "yes" dedi. Decision Tree de benzer şekilde taban çizgisini aşamadı. KNN'in hafif üstünlüğü, yerel komşuluk yapısını kullanarak bazı "no" örneklerini yakalayabilmesinden kaynaklanıyor olabilir.

Conclusion

Bu çalışmada müşteri satın alma (Purchase değişkeninin) davranışını tahmin etmek için KNN, Decision Tree ve SVM yöntemlerini kendi bireysel veri setime uyguladım. Veri seti %72.9 oranında dengesiz olduğu için %72'lik bir taban çizgisi oluştu. Üç model de bu taban çizgisinin yakınında kaldı; en iyi sonucu k = 27 ile KNN %73.5 verdi. KNN'de 1-35 arası değerlerine bakmanın yanında accuracy grafiğiyle k seçiminde grafiği de kontrol ettim ve görselleştirdim. Decision Tree'de farklı maxdepth değerlerini karşılaştırarak overfitting başlamadan önceki en iyi derinliği 3 olarak belirledim. SVM'de cost değerlerine bakarak en iyi cost değerini cost = 1 olarak buldum. Genel bulgu, bu veride özelliklerin Purchase'ı güçlü biçimde ayırmaya yetmediği ve modellerin çoğunluk sınıfına yöneldiğidir. Yine de KNN, yerel yapıyı kullanarak taban çizgisini geçen tek yöntem olarak öne çıktı denebilir.