Bilgi Yönetim Sistemleri - Bitirme Projesi Rapor : Süleyman Oruç 190401026

Gerekli kütüphanelerin yüklenmesi

```
library(readr)
library(dplyr)
library(tidyr)
library(rpart)
library(rpart.plot)
library(caret)
library(ggplot2)
```

Random veri oluşturulması

```
veri1 <-  read.csv("veri2.csv")
veri2 <-  read.csv("veri3.csv")
phone <- read.csv("phone.csv")
info <- read.csv("info.csv")



Clientnum <- veri2[sample(nrow(veri2), 5000), c("CLIENTNUM")]
info_sample <- info[sample(nrow(info), 5000), c("first_name", "last_name", "email")]
phone <- phone[sample(nrow(phone), 5000), "Phone"]
BirthDay <- veri1[sample(nrow(veri1), 5000), c("CustomerDOB")]
TransactionDate <- veri1[sample(nrow(veri1), 5000), c("TransactionDate")]
age <- veri2[sample(nrow(veri2), 5000), c("Customer_Age", "Marital_Status")]
veri2_sample <- veri2[sample(nrow(veri2), 5000), c("Dependent_count","Card_Category","Income_Category",

# Birleştir
birlestirilmis_veri <- cbind(Clientnum, info_sample, phone, BirthDay, age, TransactionDate, veri2_sampl

#Eksik değerleri NA olarak güncelle
birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Marital_Status = ifelse(Marital_Status == "Unknown", NA, Marital_Status))

birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Income_Category = ifelse(Income_Category == "Unknown", NA, Income_Category))

birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(BirthDay = ifelse(BirthDay == "nan", NA, BirthDay))

# NA değerlerinin bulunduğu satırları silme
tamveri <- birlestirilmis_veri %>%
  filter(!is.na(Income_Category))
```

```r
#Income_Category sütununununda ki verilerin character tipinden numeric tipine dönüştürülmesi sırasında ha
#Hatanın neden çözüldüğü bilinmemektedir.
tamveri <- head(tamveri, 5000)

#Income_Category sütununununda ki verilerin character tipinden numeric tipine dönüştürülmesi

set.seed(123) # Rastgeleliği tekrarlanabilir kılmak için

tamveri <- tamveri %>%
  mutate(Income_Category = case_when(
    Income_Category == "Less than $40K" ~ sample(10000:40000, 1),
    Income_Category == "$40K - $60K" ~ sample(40000:60000, 1),
    Income_Category == "$60K - $80K" ~ sample(60000:80000, 1),
    Income_Category == "$80K - $120K" ~ sample(80000:120000, 1),
    Income_Category == "$120K +" ~ sample(120000:200000, 1),
    TRUE ~ as.numeric(Income_Category)  # Diğer durumlar için mevcut değeri korur
  ))

#Income_Category sütununun
mean_income <- mean(tamveri$Income_Category)

#Veri setinde Income_Category sütununda bulunan Nan değerlerin o sütunun ortalama değeri ile doldurulma
birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Income_Category = ifelse(is.na(Income_Category), mean_income, Income_Category))

birlestirilmis_veri <- head(birlestirilmis_veri, 5000)

#Income_Category sütununununda ki verilerin character tipinden numeric tipine dönüştürülmesi
birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Income_Category = case_when(
    Income_Category == "Less than $40K" ~ sample(10000:40000, 1),
    Income_Category == "$40K - $60K" ~ sample(40000:60000, 1),
    Income_Category == "$60K - $80K" ~ sample(60000:80000, 1),
    Income_Category == "$80K - $120K" ~ sample(80000:120000, 1),
    Income_Category == "$120K +" ~ sample(120000:200000, 1),
    TRUE ~ as.numeric(Income_Category)  # Diğer durumlar için mevcut değeri korur
  ))

# Marital_Status sütununda ki verilerin anlamlı hale getirilmesi

birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Marital_Status = ifelse(Marital_Status == "Single", "Bekar", Marital_Status))

birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Marital_Status = ifelse(Marital_Status == "Married", "Evli", Marital_Status))

birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Marital_Status = ifelse(Marital_Status == "Divorced", "Bosanmis", Marital_Status))

#Card_Category sütununda ki verilerin anlamlı hale getirilmesi

birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Card_Category = ifelse(Card_Category == "Blue", "Temel", Card_Category))
```

```r
birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Card_Category = ifelse(Card_Category == "Silver", "Orta", Card_Category))

birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Card_Category = ifelse(Card_Category == "Gold", "İleri", Card_Category))

# Sütun adlarının değiştirilmesi
birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Gelir = Income_Category)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Musteri_Numarasi = Clientnum)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Isim = first_name)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Soyisim = last_name)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Telefon_numarasi = phone)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Dogum_gunu = BirthDay)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Yas = Customer_Age)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Medeni_durum = Marital_Status)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Islem_tarihi = TransactionDate)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Hesap_sayisi = Dependent_count)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Kart_kategori = Card_Category)

birlestirilmis_veri <- birlestirilmis_veri %>%
  rename(Kredi_limiti = Credit_Limit)
```

Makine öğrenmesi modeli için rastgele oluşturulan verilere aşağıda ki kod öbeğinde "ifelse" koşulları içinde belirtilen özelliklere göre Kredi Onayının olup olmadığını gösteren 1 (Kredi Onaylandı), 0 (Kredi Onaylanmadı) eklenmiştir.

Oluşturulan veri setine Kredi Onay sütununun eklenmesi

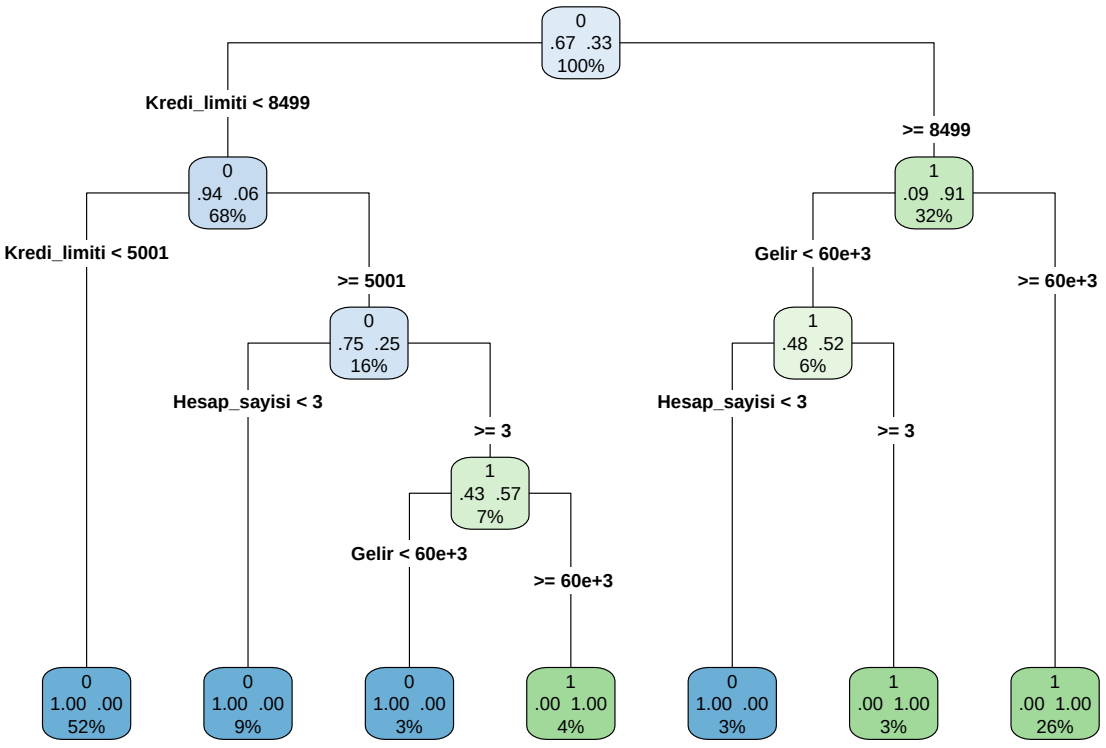
```r
birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Kredi_uygun = ifelse(Hesap_sayisi > 2 & Gelir > 63000 & Kredi_limiti > 8500, 1, 0))

birlestirilmis_veri <- birlestirilmis_veri %>%
```

```
  mutate(Kredi_uygun = ifelse(Hesap_sayisi <= 2 & Gelir > 63000 & Kredi_limiti > 8500, 1, Kredi_uygun))

birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Kredi_uygun = ifelse(Hesap_sayisi > 2 & Gelir > 30000 & Kredi_limiti > 8500, 1, Kredi_uygun))

birlestirilmis_veri <- birlestirilmis_veri %>%
  mutate(Kredi_uygun = ifelse(Hesap_sayisi > 2 & Gelir > 63000 & Kredi_limiti > 5000, 1, Kredi_uygun))
```

Makine öğrenmesi modelinin oluşturulması konusunda Decision Tree kullanılması kararlaştırılmıştır. Aşağıda gerekli kod öbekleri görüntülenmektedir.

Makine öğrenmesi modelinin oluşturulması

```
set.seed(123)
train_index <- createDataPartition(birlestirilmis_veri$Kredi_uygun, p = 0.8, list = FALSE)
train_data <- birlestirilmis_veri[train_index, ]
test_data <- birlestirilmis_veri[-train_index, ]

# Karar ağacı modeli oluşturma
kredi_uygun_tree <- rpart(Kredi_uygun ~ Hesap_sayisi + Gelir + Kredi_limiti, data = train_data, method =

# Modeli görselleştirme
rpart.plot(kredi_uygun_tree, type = 4, extra = 104)
```



```r
test_veri <-  read.csv("verib.csv")

# Test veri seti üzerinde tahmin yapma
test_pred <- predict(kredi_uygun_tree, test_veri, type = "class")



tahminler <- predict(kredi_uygun_tree, test_veri, type = "class")

# Tahminleri test veri setine ekleme
test_veri <- test_veri %>%
  mutate(Kredi_uygun = tahminler)
```

Modelin performansının değerlendirilmesi

```r
# Karışıklık Matrisi (Confusion Matrix) oluşturma
conf_matrix <- confusionMatrix(test_pred, factor(birlestirilmis_veri$Kredi_uygun))

# Performans metriklerini çıkarma
accuracy <- conf_matrix$overall["Accuracy"]
sensitivity <- conf_matrix$byClass["Sensitivity"]
specificity <- conf_matrix$byClass["Specificity"]
f1_score <- 2 * (sensitivity * specificity) / (sensitivity + specificity)

# Performans metriklerini bir data frame'e ekleme
performance_metrics <- data.frame(
  Metric = c("Accuracy", "Sensitivity", "Specificity", "F1 Score"),
  Value = c(accuracy, sensitivity, specificity, f1_score)
)

# Sonuçları gösterme
print(performance_metrics)
```

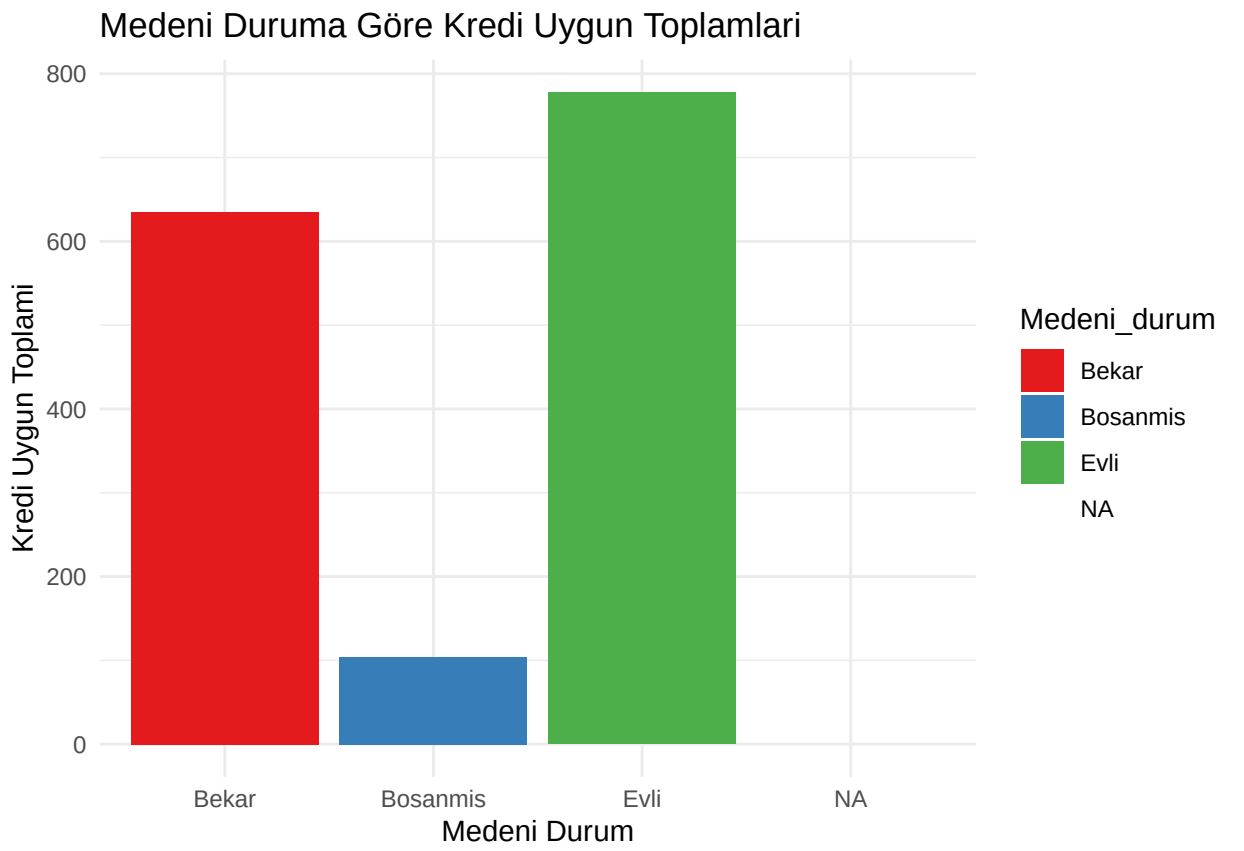
```
##        Metric     Value
## 1    Accuracy 0.5514000
## 2 Sensitivity 0.6702445
## 3 Specificity 0.3092345
## 4    F1 Score 0.4232101
```

Yukarıda ki model performans metrikleri incelendiğinde kullandığımız makine öğrenmesi tahmin modelimiz düşük bir başarı oranına sahiptir ancak bu projede yapılacak analiz için performansı yeterli görülmüştür.

Analiz sonuçlarının görselleştirilmesi

```r
# Medeni_durum'a göre gruplandırarak Kredi_uygun toplamını hesaplama
kredi_uygun_toplam <- birlestirilmis_veri %>%
  group_by(Medeni_durum) %>%
  summarise(Kredi_uygun_toplam = sum(Kredi_uygun))

# Grafiği oluşturma
ggplot(kredi_uygun_toplam, aes(x = Medeni_durum, y = Kredi_uygun_toplam, fill = Medeni_durum)) +
  geom_bar(stat = "identity") +
  labs(title = "Medeni Duruma Göre Kredi Uygun Toplamları",
```

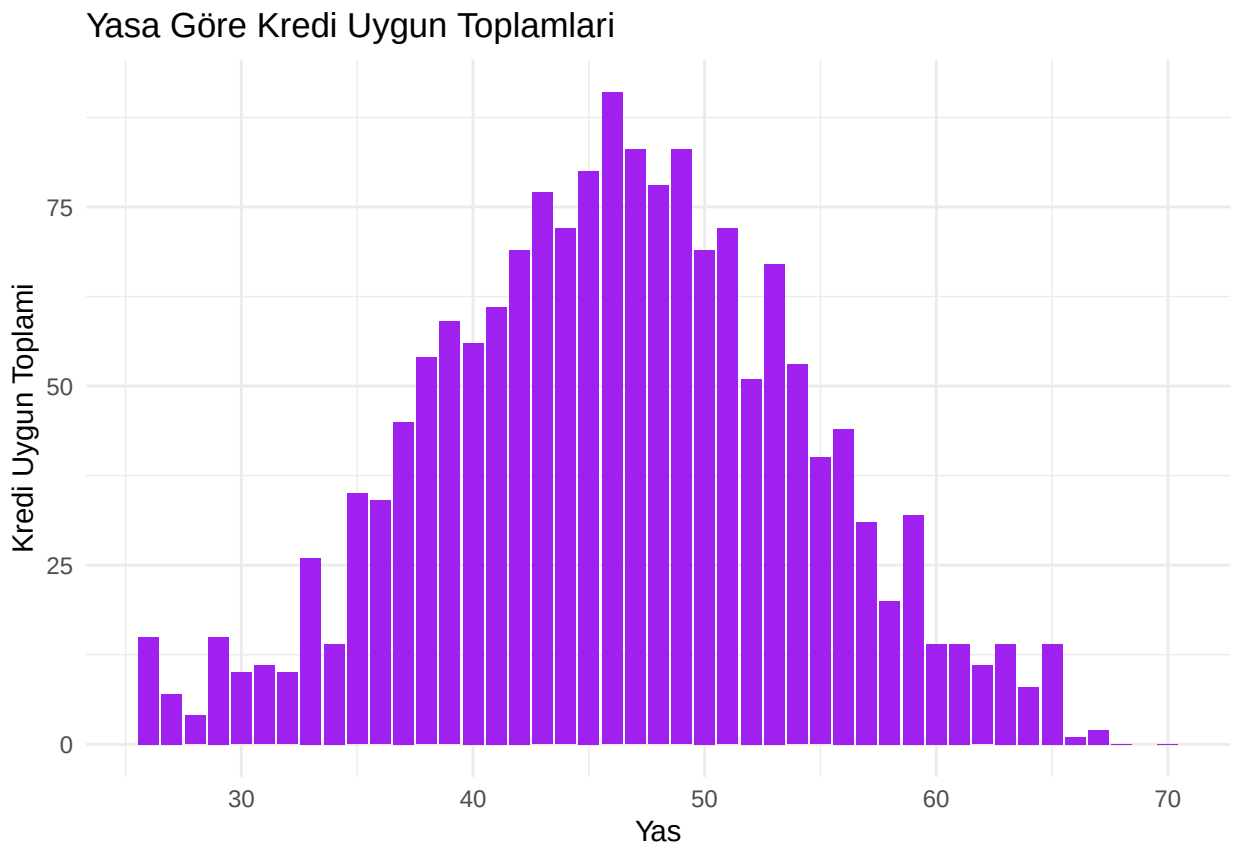
```
       x = "Medeni Durum",
       y = "Kredi Uygun Toplamı") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set1")
```



Yukarıda ki grafikte yapılan analizde Kredi Onayı alan müşterilerin medeni durumları ele alınmıştır. Grafik incelendiğinde evli ve bekar kişilerde pek fark gözlemlenmezken boşanmış kişilerin kredi açısından riskli grupta yer aldığı değerlendirilmiştir.

Analiz sonuçlarının görselleştirilmesi

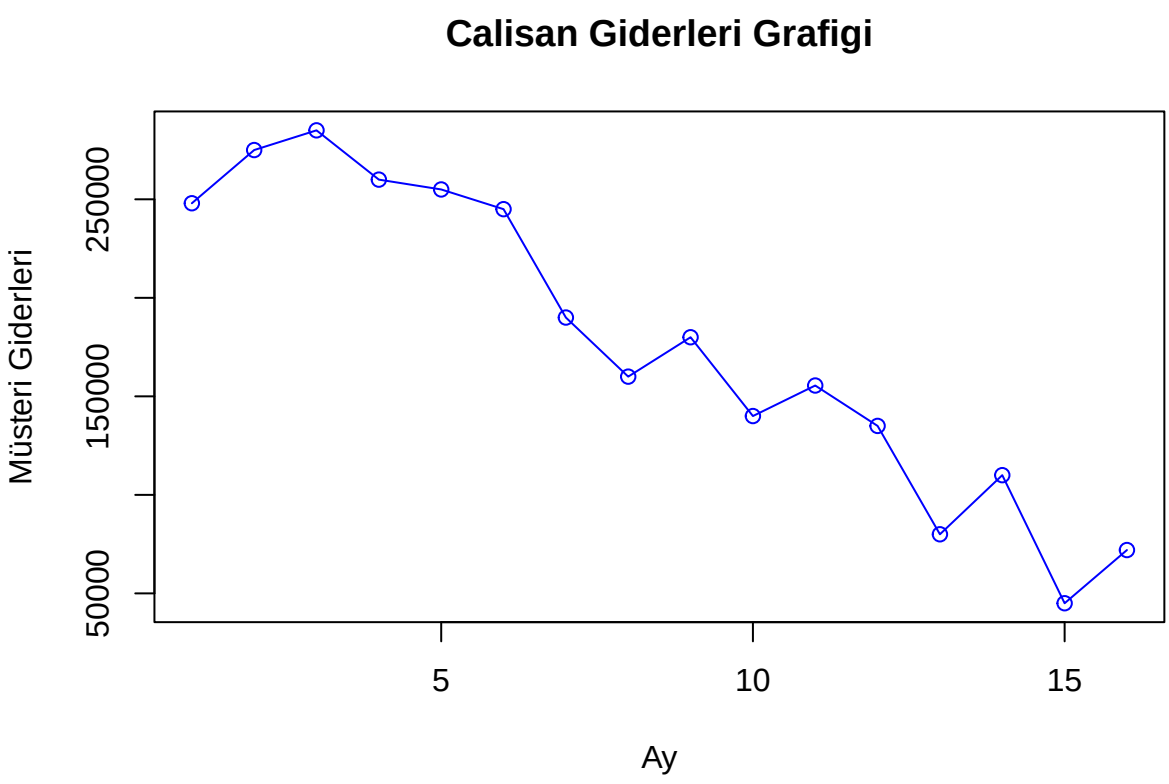
```
kredi_uygun_toplam <- birlestirilmis_veri %>%
  group_by(Yas) %>%
  summarise(Kredi_uygun_toplam = sum(Kredi_uygun))

# Grafiği oluşturma
ggplot(kredi_uygun_toplam, aes(x = Yas, y = Kredi_uygun_toplam)) +
  geom_bar(stat = "identity", fill = "purple") +
  labs(title = "Yasa Göre Kredi Uygun Toplamları",
       x = "Yaş",
       y = "Kredi Uygun Toplamı") +
  theme_minimal()
```



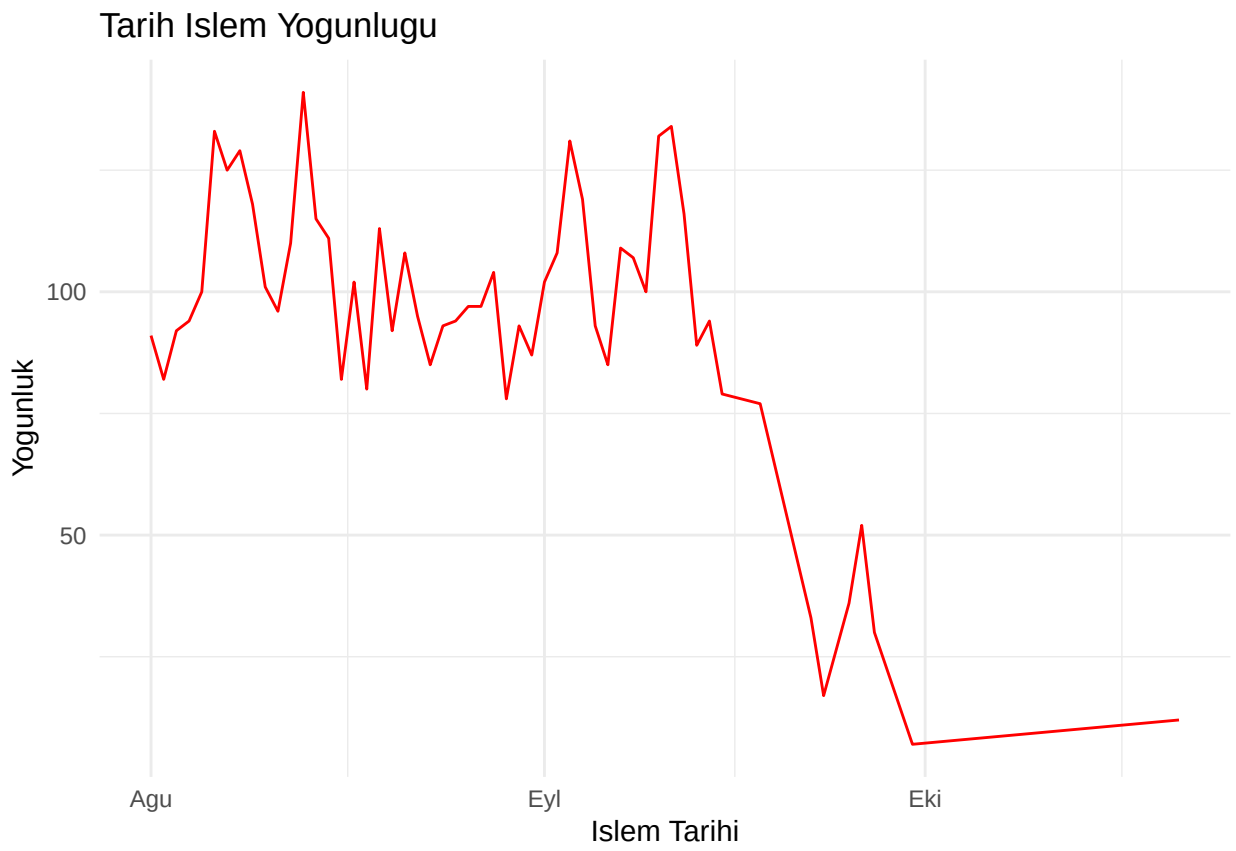
Yukarıda ki grafikte Kredi Onayı alan kişilerin yaş dağılımları gösterilmektedir. Bu grafikte belirtilen analiz doğrultusunda 30'lu yaş altı ve 60 yaş üstü müşterilerin kredi onayı açısından riskli yaş gruplarında olduğu değerlendirilmiştir.

Analiz sonuçlarının görselleştirilmesi

```
birlestirilmis_veri$Islem_tarihi <- as.Date(birlestirilmis_veri$Islem_tarihi, format = "%d/%m/%y")


sorted_data <- birlestirilmis_veri[order(birlestirilmis_veri$Islem_tarihi), ]


tarih_yogunlugu <- sorted_data %>%
  group_by(Islem_tarihi) %>%
  summarise(yogunluk = n())

# Veri çerçevesini oluşturma
calisan_giderleri <- data.frame(
  giderler = c(248000, 275000, 285000, 260000, 255000, 245000, 190000, 160000, 180000, 140000, 155500,
)
plot(calisan_giderleri$giderler, type = "o", col = "blue",
     xlab = "Ay", ylab = "Müşteri Giderleri",
     main = "Calisan Giderleri Grafiği")
```



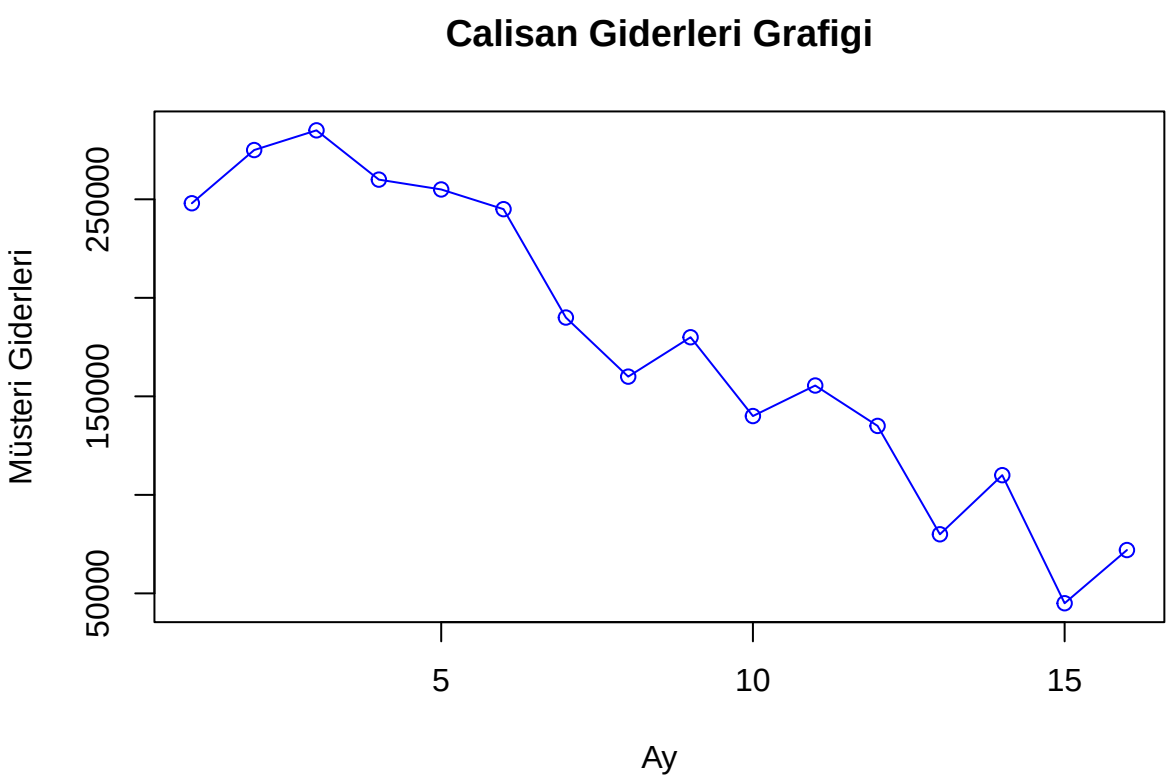

```r
musteri <- test_data$Musteri_Numarasi
isim <- test_data$Isim
soyisim <- test_data$Soyisim
kredi <- test_data$Kredi_uygun

son <- cbind(musteri,isim,soyisim,kredi)

data <- c(1:54)  # Örnek olarak 1'den 54'e kadar olan sayıları kullanıyoruz

# Tek sütunlu bir data frame oluşturuyoruz
df <- data.frame(column1 = data)

tarih <- cbind(tarih_yogunlugu,df)
write.csv(son, file = "son.csv", row.names = FALSE)
write.csv(tarih, file = "tarih.csv", row.names = FALSE)

ggplot(data = tarih_yogunlugu, aes(x = Islem_tarihi, y = yogunluk)) +
  geom_line(color = "red") +  # Çizgi grafiği çizimi, rengi kırmızı olarak belirledik
  labs(title = "Tarih İşlem Yoğunluğu",
       x = "İşlem Tarihi",
       y = "Yoğunluk") +  # Grafik için başlık ve eksen etiketlerini ekliyoruz
  theme_minimal()  # Temayı minimal yapıyoruz
```



Yukarıda ki grafikte bankanın günlere göre işlem sayısı gösterilmektedir. Bu analiz grafiği kullanılarak banka çalışanlarının günlük yoğunluğa göre belirlenip çalışan giderlerinde tasarrufa gidilmesi öngörülmektedir.

Analiz sonuçlarının görselleştirilmesi

```
# Veri çerçevesini oluşturma
calisan_giderleri <- data.frame(
  giderler = c(248000, 275000, 285000, 260000, 255000, 245000, 190000, 160000, 180000, 140000, 155500,
)
plot(calisan_giderleri$giderler, type = "o", col = "blue",
     xlab = "Ay", ylab = "Müşteri Giderleri",
     main = "Calisan Giderleri Grafiği")
```



Yukarıda ki grafikte 6.ayda yapılan işlem yoğunluğu analizine göre çalışan sayısının ayarlanmasından sonra bankanın çalışan giderlerinin değişimi gösterilmektedir. Grafikte de görüldüğü üzere işlem yoğunluğu analizi bankamızın tasarrufu için önemli bir adım olmuştur.

Projenin canlıya alınması

OvhCloud Sunucu Üzerinde SuiteCRM Kurulumu

Giriş

Bu rapor, işletmemizde kullanılan Bilgi Yönetim Sistemi (BYS) kapsamında OverHead (ovhd) sunucu kiralayarak SuiteCRM'nin nasıl kurulduğunu detaylı bir şekilde anlatmaktadır. SuiteCRM, açık kaynaklı bir CRM yazılımıdır ve bu kurulum süreci, mevcut sistemlerimizi geliştirmek ve müşteri ilişkilerini daha etkin bir şekilde yönetmek için gerçekleştirilmiştir.

1. Gereksinimler

SuiteCRM kurulumuna başlamadan önce gerekli olan bileşenler ve araçlar şunlardır: • Ovhcloud hesabı • SSH erişimi • Docker kurulu bir web server • SuiteCRM kurulum dosyaları

2. OvhCloud Sunucu Kiralama 1. Hesap Oluşturma: OverHead web sitesine giderek bir hesap oluşturduk. 2. Sunucu Seçimi: İş gereksinimlerimize uygun bir sunucu planı seçtik. SuiteCRM'nin çalışması için önerilen sistem gereksinimlerine uygun bir plan belirledik. 3. Sunucu Kurulumu: Seçilen sunucuyu oluşturduk ve gerekli bilgileri aldık (IP adresi, kullanıcı adı, şifre).

3 .Sunucuya SuiteCRM kurup Bağlantıları Sağlamak

Suite imagesini indirin

docker pull bitnami/suitecrm:latest

Bağlantı oluşturun

docker network create suitecrm-network

MariaDB kalıcılığı için bir hacim oluşturun ve bir MariaDB konteyneri oluşturun

docker volume create –name mariadb_data docker run -d –name mariadb
–env ALLOW_EMPTY_PASSWORD=yes
–env MARIADB_USER=bn_suitecrm
–env MARIADB_PASSWORD=bitnami
–env MARIADB_DATABASE=bitnami_suitecrm
–network suitecrm-network
–volume mariadb_data:/bitnami/mariadb
bitnami/mariadb:latest

SuiteCRM kalıcılığı için hacimler oluşturun ve konteyneri başlatın

docker volume create –name suitecrm_data docker run -d –name suitecrm
-p 8080:8080 -p 8443:8443
–env ALLOW_EMPTY_PASSWORD=yes
–env SUITECRM_DATABASE_USER=bn_suitecrm
–env SUITECRM_DATABASE_PASSWORD=bitnami
–env SUITECRM_DATABASE_NAME=bitnami_suitecrm
–network suitecrm-network
–volume suitecrm_data:/bitnami/suitecrm
bitnami/suitecrm:latest

oluşturduğumuz web bağlantısı

http://162.19.249.190:8080/