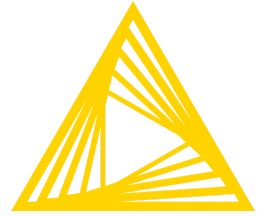


Kaggle - Titanik Yarışması

Büyük Veri Araçlarını kullanarak Titanik Felaketinden kurtulan sayısını tahmin etmek.



kaggle™



Onur Osman Güle
G171210021

Fatih Enis Kaya
G171210375

Titanik Yarışması Hakkında

15 Nisan 1912 tarihinde Titanik gemisi bir buz dağına çarptığında bir felaket meydana geldi. Bu felaketin Oscar ödüllü bir filmi de çekilmiştir.

Bu yarışmada veri setleri verilerek hangi özelliklere sahip olan yolcuların hayatta kalma olasılıklarını bulmamız gerekiyor, hangi yolcu hayatta, hangisi hayatını kaybetmiş diye istatistik çıkarmamız gerekiyor.

Veriyi İşlemek

Kaggle bize 2 adet CSV dosyası veriyor bize.

1. *train.csv*: Modelimizi oluşturmak için gereken sütunların mevcut olduğu veri setimiz.
2. *test.csv*: Modelimizi test etmek için kullanacağımız veri setimiz.

train.csv veri setimizde şu sütunlar mevcuttur:

- *Survived*: Hayatta Kalma
- *Pclass*: Bilet Sınıfı
- *Sex*: Cinsiyet
- *Sibsp*: Titanik'teki kardeş/eş sayısı
- *Parch*: Titanik'teki ebeveyn/çocuk sayısı
- *Ticket*: Bilet No
- *Fare*: Bilet Ücreti
- *Cabin*: Kabin No
- *Embarked*: Biniş Limanı

Veri setlerinde hayatta kalanları hangi sütunlara göre hayatta kaldığını ölçmeliyiz.

Veri İşleme Platformu

Biz veri işlerken 2 ayrı yöntem(platform) kullanacağız ve ikisini karşılaştıracacağız:

1. Python
2. KNIME

İlk Yöntem: Python

Python


Python, nesne yönelimli, yorumlamalı, birimsel ve etkileşimli yüksek seviyeli bir programlama dilidir. Kaggle platformunda R veya Python kullanabiliriz. Python, daha önce programlama dili kullananlar için daha rahat bir geliştirme ortamı sunduğundan Python kullanmaya karar verdik.

Notebook Oluşturma


Python ile çalışırken Kaggle platformunda notebook üzerinden çalışmayı uygun gördük.


Select new notebook settings


You can change these settings at any time

 Select language

Python

 Select type

☒  **Notebook**
Ideal for interactive data exploration and polished analysis. Shares insights through code & commentary

☐  **Script**
Ideal for fitting a model and competition submissions. Shares code for review and RMarkdown reports

✓ SHOW ADVANCED SETTINGS

Create

Gerekli Kütüphaneler

Challenge'ı çözerken kullanacağımız yöntemlerin gerektirdiği kütüphaneleri import ettik.

```
In [1]:
import time
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from keras import models
from keras.layers import Dense, Dropout
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
#gerekli kutuphaneler eklendi
```

Veri Setlerini Okuma

Test veri seti(test.csv) ve Eğitim veri setini(train.csv) okuyoruz.

```
In [2]:
#Test veri setini okuyoruz.
tnc_test = pd.read_csv('../input/titanic/test.csv')
#Eğitim veri setini okuyoruz.
tnc = pd.read_csv('../input/titanic/train.csv')
```

Eksik Değerleri Bulma

Eksik değerleri doldurmamız gerekiyor, onları tespit etmek için missingdata isimli fonksiyon oluşturup her iki veri setindeki eksik değerleri tespit ediyoruz.

```
In [3]:
#eksik bilgi içeren sütunları doldurmak için tespit etmek
def missingdata(data):
    total = data.isnull().sum().sort_values(ascending = False)
    percent = (data.isnull().sum()/data.isnull().count()*100).sort_values(ascending = False)
    ms=pd.concat([total, percent], axis=1, keys=['Toplam', 'Yüzde'])
    ms= ms[ms["Yüzde"] > 0]
    f,ax =plt.subplots(figsize=(8,6))
    plt.xticks(rotation='90')
    fig=sns.barplot(ms.index, ms["Yüzde"],color="green",alpha=0.8)
    plt.xlabel('Sütunlar', fontsize=15)
    plt.ylabel('Eksik Sütun Yüzdesi', fontsize=15)
    plt.title('Eksik Sütunlar', fontsize=15)
    return ms
#eğitim veri setindeki eksik değerler olan sütunlar
tnc.columns[tnc.isna().any()]
#test veri setindeki eksik değerler olan sütunlar
tnc_test.columns[tnc_test.isna().any()]

missingdata(tnc)
```

Out[3]:

	Toplam	Yüzde
Cabin	687	77.104377
Age	177	19.865320
Embarked	2	0.224467

Cabin

Age
Sütunlar

Embarked

Eksik Değerleri Doldurma ve Sayısallaştırma

Embarked, Age, Fare sütunlarında eksik değer olduğunu gördük, Embarked'de en sık kullanılanı, Age ve Fare'de ortalama değerleri doldurduk.

Yaş, Kardeş Sayısı, Ebeveyn ve Çocuk Sayısı ve Bilet Fiyat verilerini 0 ile 1 arası ölçeklendirdik.

Cinsiyet sütununu ise 0 ve 1 olarak sayısallaştırdık.

```
In [5]: def eksikSutunDoldur(tnc):
# Eksik değerler yerine ortalama olarak en çok kullanılanları yazdık.
tnc['Embarked'].fillna(tnc['Embarked'].mode()[0], inplace=True)
# Eksik yaşlara ortalama yaş
tnc['Age'].fillna(tnc['Age'].median(), inplace=True)
# Eksik fiyatlara ort. fiyatı yazıldı
tnc['Fare'].fillna(tnc['Fare'].median(), inplace=True)

# Yaş, Kardeş Sayısı, Ebeveyn ve Çocuk Sayısı ve Bilet Fiyat verilerini 0 ile 1 arası ölçekle
ndirdik.
scaler = MinMaxScaler()
tnc[['Age', 'SibSp', 'Parch', 'Fare']] = scaler.fit_transform(tnc[['Age', 'SibSp', 'Parch', 'Far
e']])

# Cinsiyetleri daha verimlilik açısından 0 ve 1 ile yazdık.
tnc['Sex'] = tnc['Sex'].map({'female':0, 'male':1}).astype(int)

#One Hot Encoding(OHE): Sütunları bölüp 1,0 matrisiyle yazmak.

# Class sütununa OHE uyguladık.
tnc_class = pd.get_dummies(tnc['Pclass'], prefix='Class')
tnc[tnc_class.columns] = tnc_class

# Biniş limanına OHE uyguladık.
tnc_emb = pd.get_dummies(tnc['Embarked'], prefix='Emb')
tnc[tnc_emb.columns] = tnc_emb

# İsimleri cinsiyetleştirdik.
tnc['Title'] = tnc['Name'].map(lambda x: getName(x))
tnc['Title'] = tnc.apply(replaceName, axis=1)
# Son isimlere de OHE uyguladık.
tnc_title = pd.get_dummies(tnc['Title'], prefix='Title')
tnc[tnc_title.columns] = tnc_title

return
```

Sütun Filtreleme

Hayatta kalmaya etki edebilecek sütunları belirleyip bir boyut olarak aldık(X), hayatta kalmayı da diğer boyut olarak aldık(Y).

```
In [6]: eksikSutunDoldur(tnc)
tnc.columns #Son sütun değerlerimiz içerisinde kurtulma olasılığını etkileyebilecek sütunları belirledik.
#Modelde kullanabileceğimiz sütunlar aşağıdaki gibidir:
columns = ['Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Class_1', 'Class_2',
           'Class_3', 'Emb_C', 'Emb_Q', 'Emb_S', 'Title_Master',
           'Title_Miss', 'Title_Mr', 'Title_Mrs']
```

```
In [7]: # Sütunları bir numpy arrayına atadık. Bunu bir dimension olarak aldık, X
X = np.array(tnc[columns])
# Hayatta kalanları da diğer dimension olarak aldık, Y
y = np.array(tnc['Survived'])
```

Sinir Ağı Oluşturma

Modelimizi oluşturmak için bir sinir ağı oluşturuyoruz.

```
In [8]: network = models.Sequential()
#Bir sinir ağı oluşturuyoruz.
network.add(Dense(64, activation='relu', ))
network.add(Dropout(rate=0.2))
network.add(Dense(32, activation='relu'))
network.add(Dropout(rate=0.2))
network.add(Dense(12, activation='relu'))
network.add(Dropout(rate=0.1))
network.add(Dense(1, activation='sigmoid'))
```

Model Derlemesi

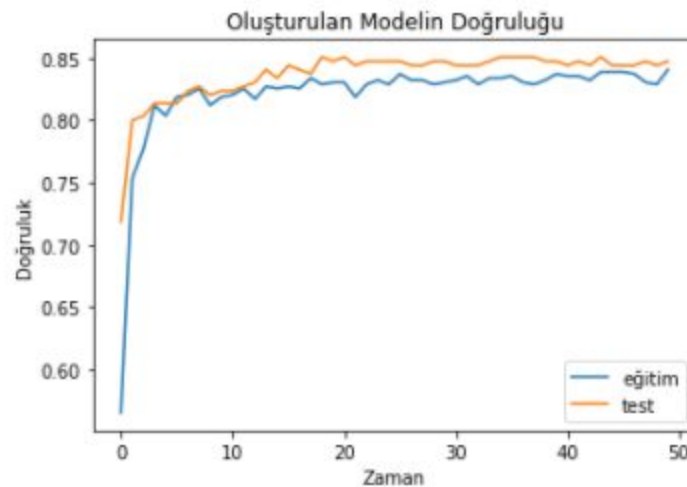
Modeli Adam optimizasyon algoritmasını kullanarak derliyoruz. İkili sınıflandırma için de `binary_crossentropy` amaç fonksiyonunu kullanıyoruz.

```
In [9]:
#Modeli Adam optimizasyon algoritmasını kullanarak derliyoruz. İkili sınıflandırma için de binary
_crossentropy amaç fonksiyonunu kullanıyoruz.
network.compile(optimizer='adam',
                loss='binary_crossentropy',
                metrics=['accuracy'])

# Geçmiş ölçümleri eğitip kaydediyoruz.
history = network.fit(X, y, epochs=50, batch_size=10, verbose=0, validation_split=0.33)

# Son çıktının grafiğini çizdiriyoruz.
plt.plot(history.history['accuracy'], label = 'eğitim')
plt.plot(history.history['val_accuracy'], label = 'test')
plt.title('Oluşturulan Modelin Doğruluğu')
plt.ylabel('Doğruluk')
plt.xlabel('Zaman')
plt.legend(loc='lower right')
plt.show()
```

Daha sonra doğruluğu ölçmek için çıktı grafiğini çizdiriyoruz.



Görüldüğü üzere oluşturduğumuz model %84 oranında doğru yanıt vermekte.

Test Veri Kontrolü

Test veri setinde de sütunları dolduruyoruz ve boyutları belirliyoruz.

```
In [10]: eksikSutunDoldur(tnc_test)

X_pred = np.array(tnc_test[columns])
y_pred = network.predict(X_pred)

y_pred = y_pred.reshape(418)

# Yolcu ID ile Hayatta Kalma oranlarını birleştirdik.
tnc_subm = pd.DataFrame({'PassengerId':tnc_test['PassengerId'], 'Survived':y_pred})
```

PassengerId ve Survived sütunlarını birleştiriyoruz.

```
In [11]: #Hatta kalma oranlarını kesin olarak tahmin için 1 ve 0'a dönüştürdük.
def binary(x):
    #Hayatta kalma oranı 0.5 üzerindeyse hayatta kaldığını düşüneceğiz.
    if x >= 0.5:
        return 1
    else:
        return 0

# Survived sütunundaki tüm değerleri binary değere dönüştürdük.
tnc_subm['Survived'] = tnc_subm['Survived'].apply(binary)
```

Hayatta kalma olasılığını, 0.5 üstüyse hayatta kalmıştır, değilse hayatta kalamamıştır olarak belirliyoruz.

Sonucu Dosyaya Yazdırma

Bulduğumuz sonuç dosyasını Submission olarak gönderebilmek için bir csv dosyasına yazdırdık.

```
In [12]:
filename = 'G171210021_G171210375_BigData'.csv
tnc_subm.to_csv(filename,index=False)

print('Sonuç Dosyası: ' + filename)

#En sonunda ID ve binary olarak kurtulanları gösteren bir csv sonuç dosyası oluşturduk.
#%82-88 arası bir sonuç aldık, elbette %100 yapanlar var ancak bunun için farklı algoritmalarla
yararlanmak gerekiyor.
#G171210021 - Onur Osman Güle
#G171210375 - Fatih Enis Kaya
#Sakarya Üniversitesi Bilgisayar Mühendisliği
```

Sonuç Dosyası: G171210021_G171210375_BigData.csv

İkinci Yöntem: KNIME

KNIME

KNIME açık kaynak ve çapraz platform veri analizi, raporlama, entegrasyon platformudur. Neredeyse hiç kod yazmadan veri analizi yapılabilen bir platformdur.

Proje Oluşturma

KNIME platformunu indirip kurduktan sonra bir proje oluşturuyoruz.

New Project

Project
Create a new project resource.

Project name:

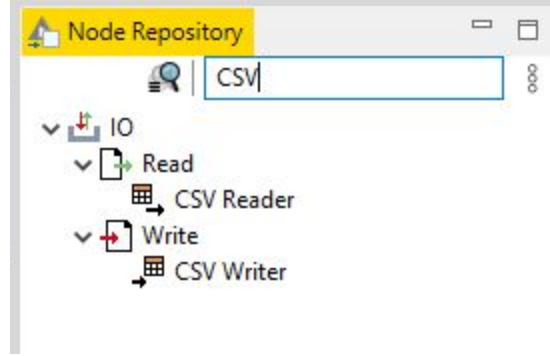
☒ Use default location
Location:

Working sets
☐ Add project to working sets
Working sets:

< Back Next > **Finish** Cancel

CSV Okuma İşlemi

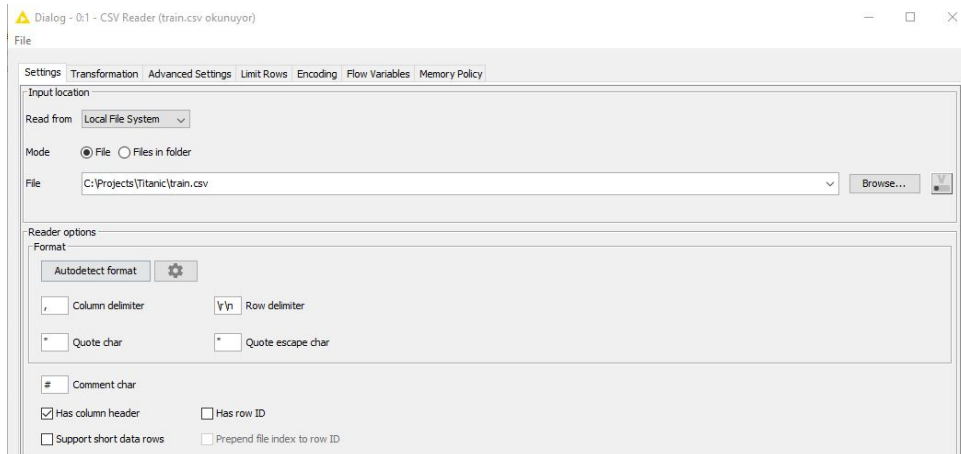
Öncelikle train.csv'nin içindekileri okumamız gerekmekte. Bunun için Node Repository içerisinde CSV Reader Node'unu kullanmamız gerekiyor.



Node Repository'den bulduktan sonra proje ekranımıza sürüklüyoruz.

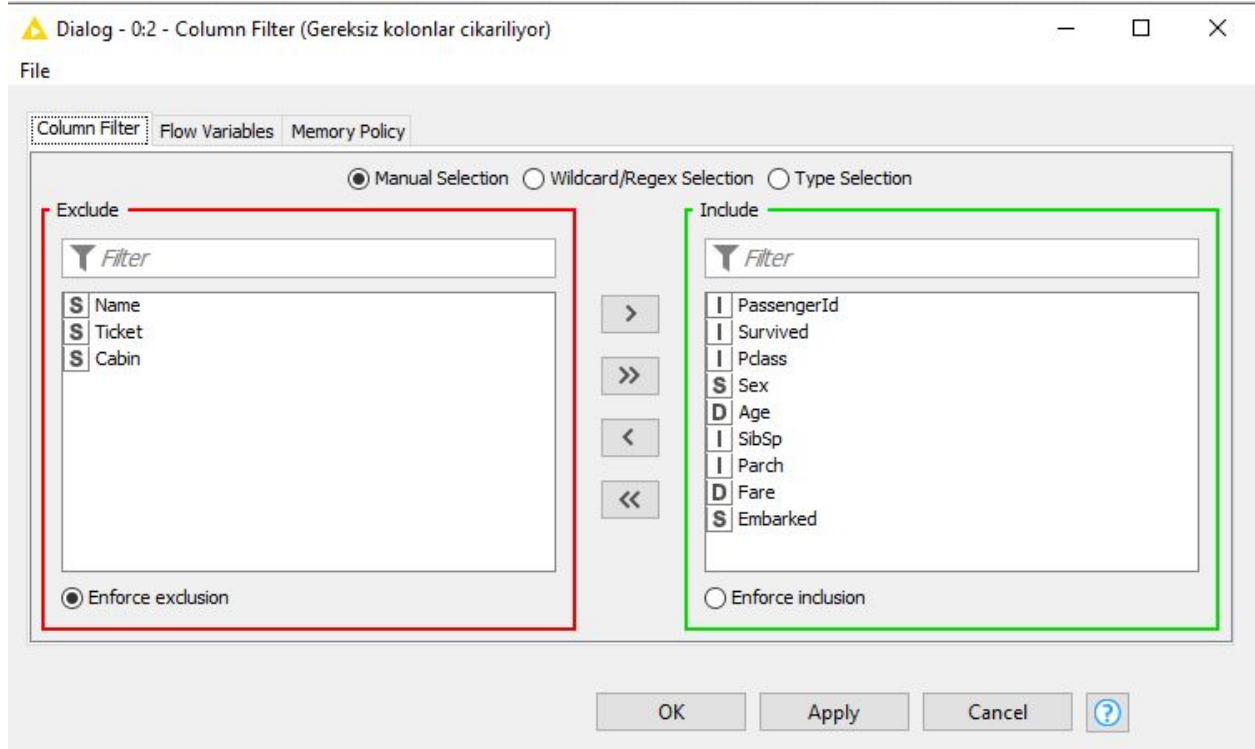


Sonrasından gerekli csv dosyasını seçip onaylıyoruz.



Sütun Filtreleme

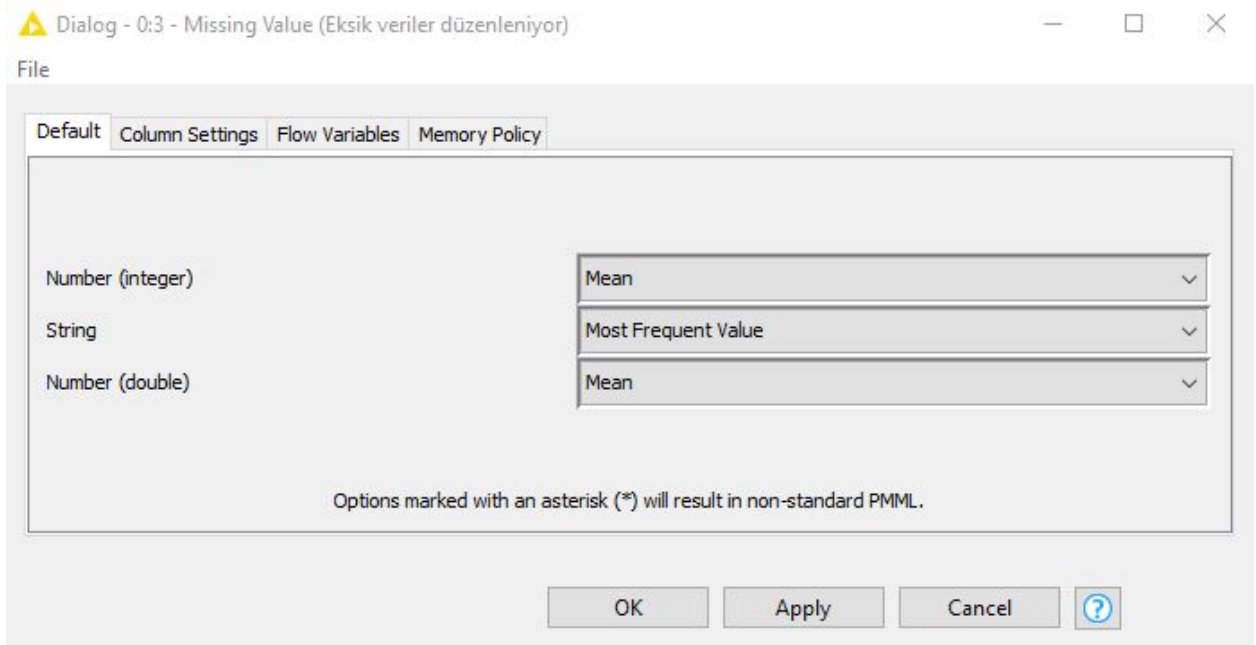
Train.csv'deki bazı sütunların yolcuların kurtulmalarıyla alakalı olmadığı için bu sütunları filtrelemeye karar verdik. Bunun için Column Filter node'unu kullanıyoruz.



Name, Ticket ve Cabin sütunları, kurtulmayla ilgili olmadığından exclude ediyoruz.

Eksik Değerleri Doldurma

Bazı sütunlarda eksik değerler mevcut. Bunları Missing Value node'u ile dolduruyoruz. Int ve double veri tipi olan değerleri ortalama, string veri tipi olan değerleri ise en çok tekrarlanan değerlerle dolduruyoruz. Elbette tamamen doğru doldurulmayacak ama bizi sonuca yaklaştıracaktır.

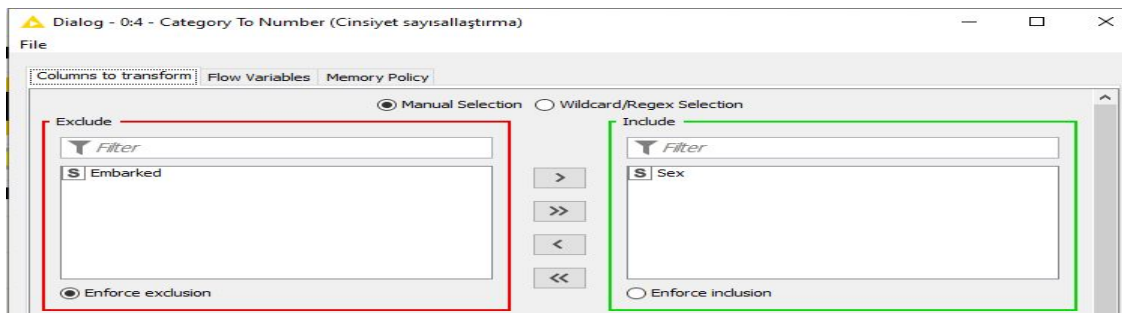


Kategori Sayısallaştırma

Bazı sütunlarda aynı değerleri birden çok kullanmış, bunları sayısallaştırmak işlemimizi kolaylaştıracaktır.

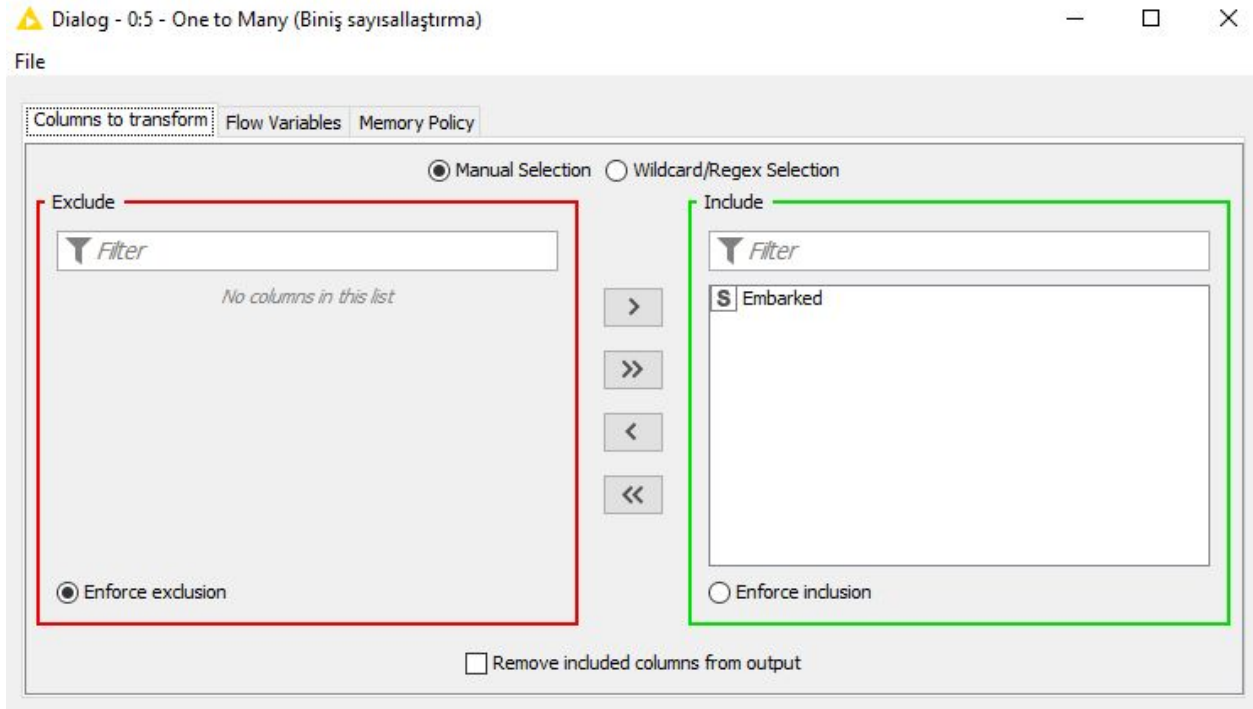
Cinsiyet Sayısallaştırma

Cinsiyet(Sex) sütunu yalnızca 2 değer almakta, bunları Category To Number kullanarak 0 ve 1 olarak değiştirelim.



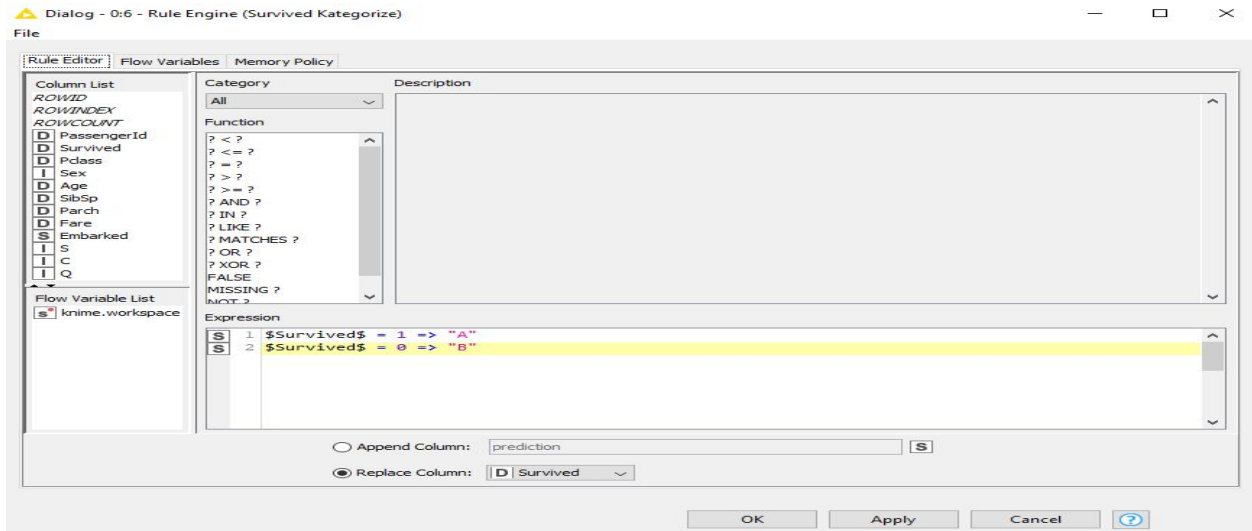
Biniş Sayısallaştırma

Embarked(Biniş) sütunu ise 2'den fazla değer aldığı için One To Many node'u ile sayısallaştıracğız.



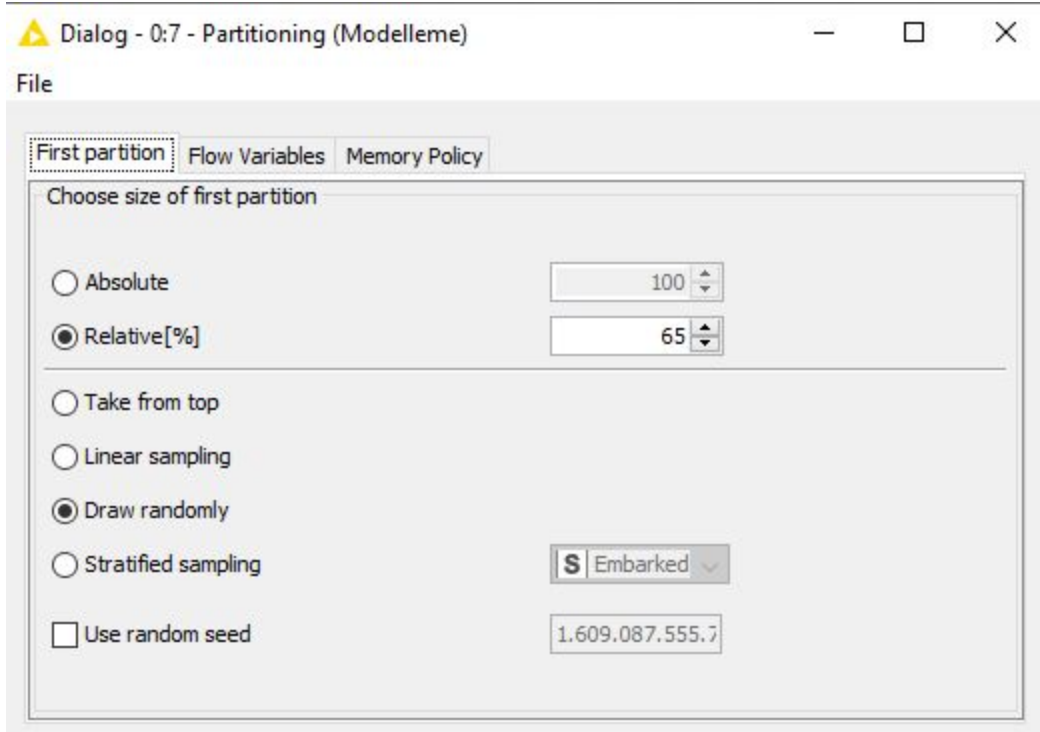
Hayatta Kalmayı Kategorizeleme

Hayatta kalanları bulmak istediğimiz için Survived sütununu Rule Engine node'u kullanarak sayısal değerden kategorik değere dönüştüreceğiz.



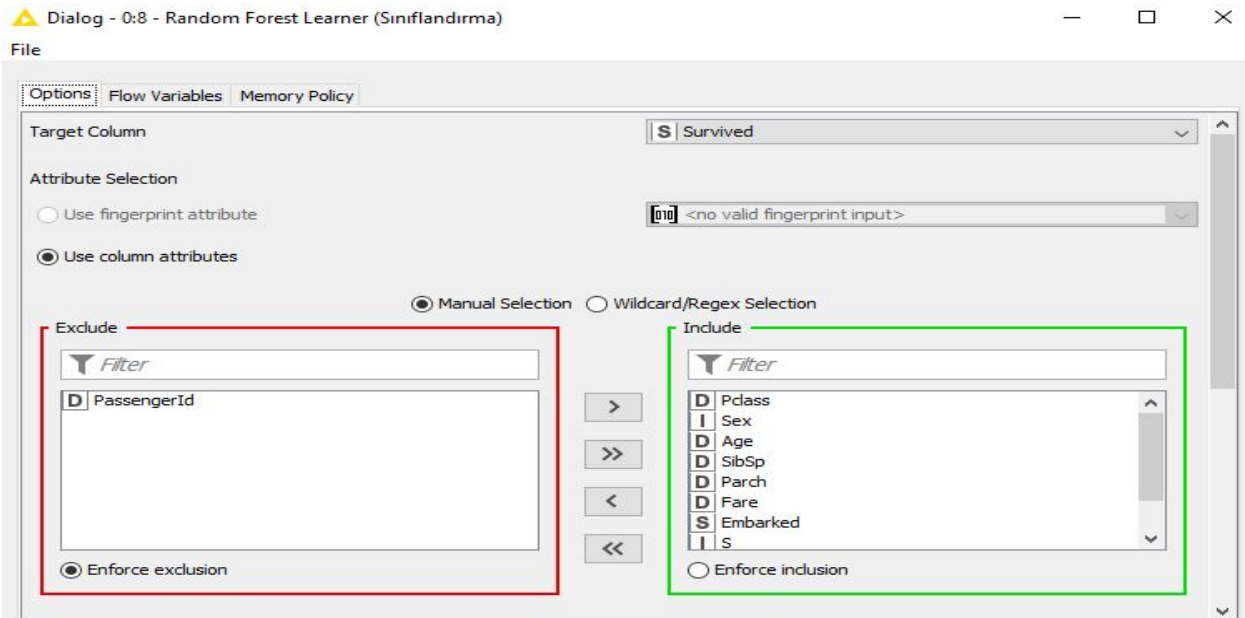
Veriyi Bölmek

Modellemeye geçmeden önce modeli train verisi ile test edebilmek için veriyi Partitioning node'u ile böleceğiz.



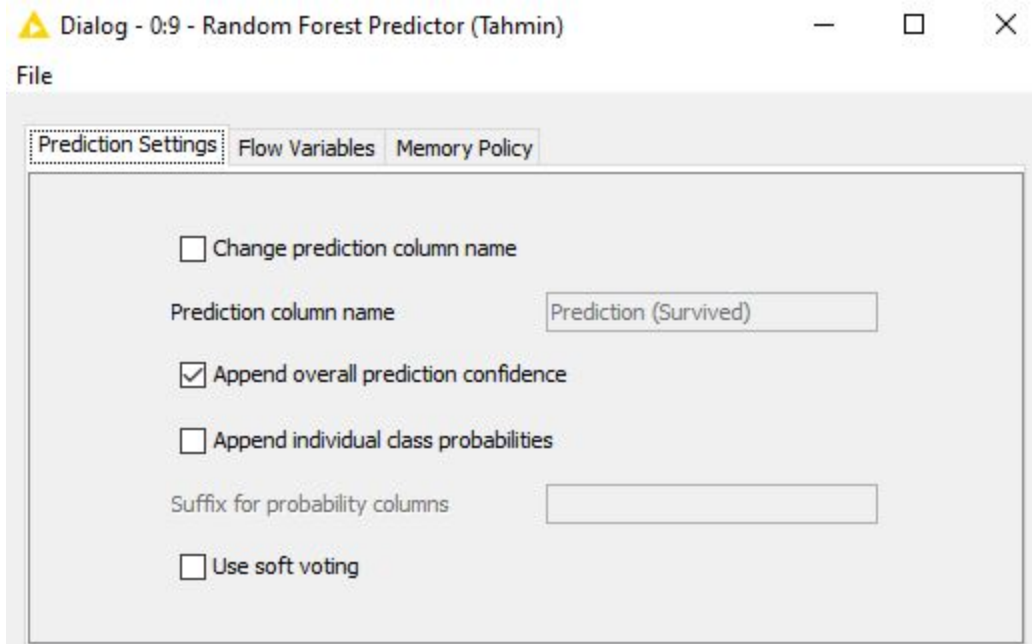
Öğrenme

Böldüğümüz verinin bir sınıflandırma algoritmasıyla öğrenme sürecini başlatacağız. Bunun için genel amaçlı bir algoritma olan Random Forest algoritmasını kullanacağız.



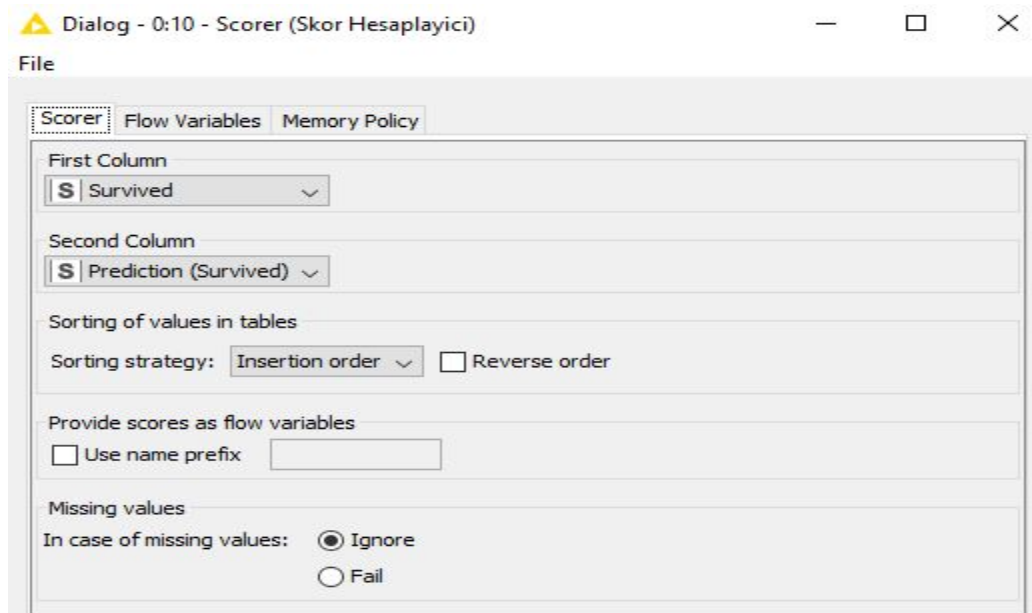
Tahmin

Modelimiz hazır, verimizin test etmek için böldüğümüz diğer kısmı alıp tahmin etmeye çalışacağız. Bunun için Random Forest Predictor node'unu kullanacağız.



Model Doğruluğu

Scorer node'unu kullanarak modelimizin doğruluğunu test edelim.



Model Skoru

Confusion matrix - 0:10 - Scorer (Skor Hesaplayıcı)

File

Table "spec_name" - Rows: 2 Spec - Columns: 2 Properties Flow Variables

Index	Owner ID	Name	Value
0	0:10	d Cohen's kappa	0.6582618025751072
0	0:10	i #False	49
0	0:10	i #Correct	263
0	0:10	d Error	0.15705128205128205
0	0:10	d Accuracy	0.842948717948718
0		s knime.workspace	C:\Users\onurg\knime-workspace

Scorer kullandıktan sonra gözlemlediğimiz üzere modelimiz %84.42 oranında doğruluk derecesinde tahmin ediyor.

Test Verisini Okumak

Yukarıda train.csv için yaptığımız işlemleri test.csv için tekrarlıyoruz.

Sonuç Çıkarımı

Yalnızca PassengerId ve Survived sütunlarını göstermek istediğimizden Column Filter kullanıyoruz.

Dialog - 0:17 - Column Filter (Sonuç Filtreleme)

File

Column Filter Flow Variables Memory Policy

☒ Manual Selection ☐ Wildcard/Regex Selection ☐ Type Selection

Exclude

Filter

- D Pclass
- I Sex
- D Age
- D SibSp
- D Parch
- D Fare
- S Embarked
- I Q
- I S
- I C
- D Survived (Confidence)

☒ Enforce exclusion

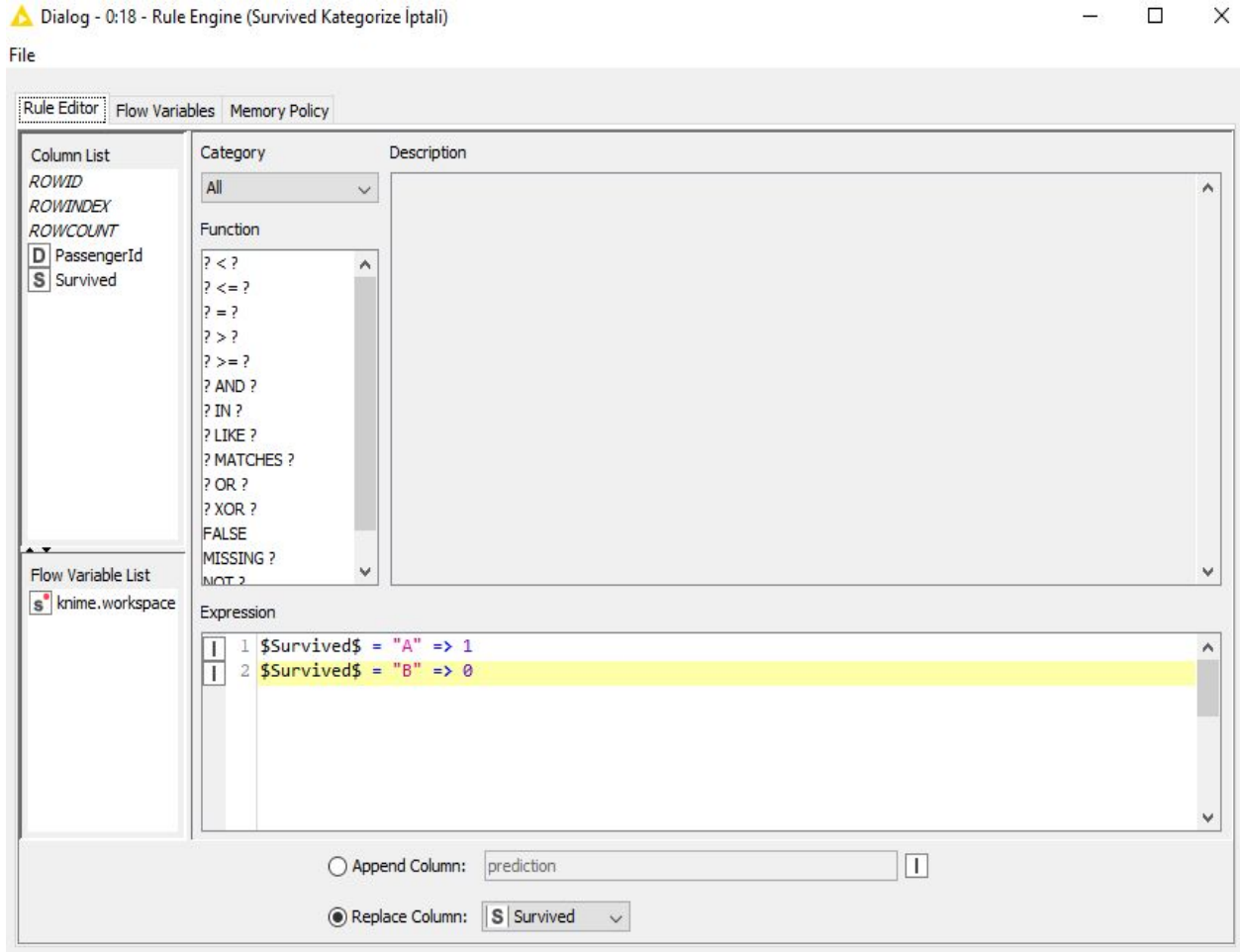
Include

Filter

- D PassengerId
- S Survived

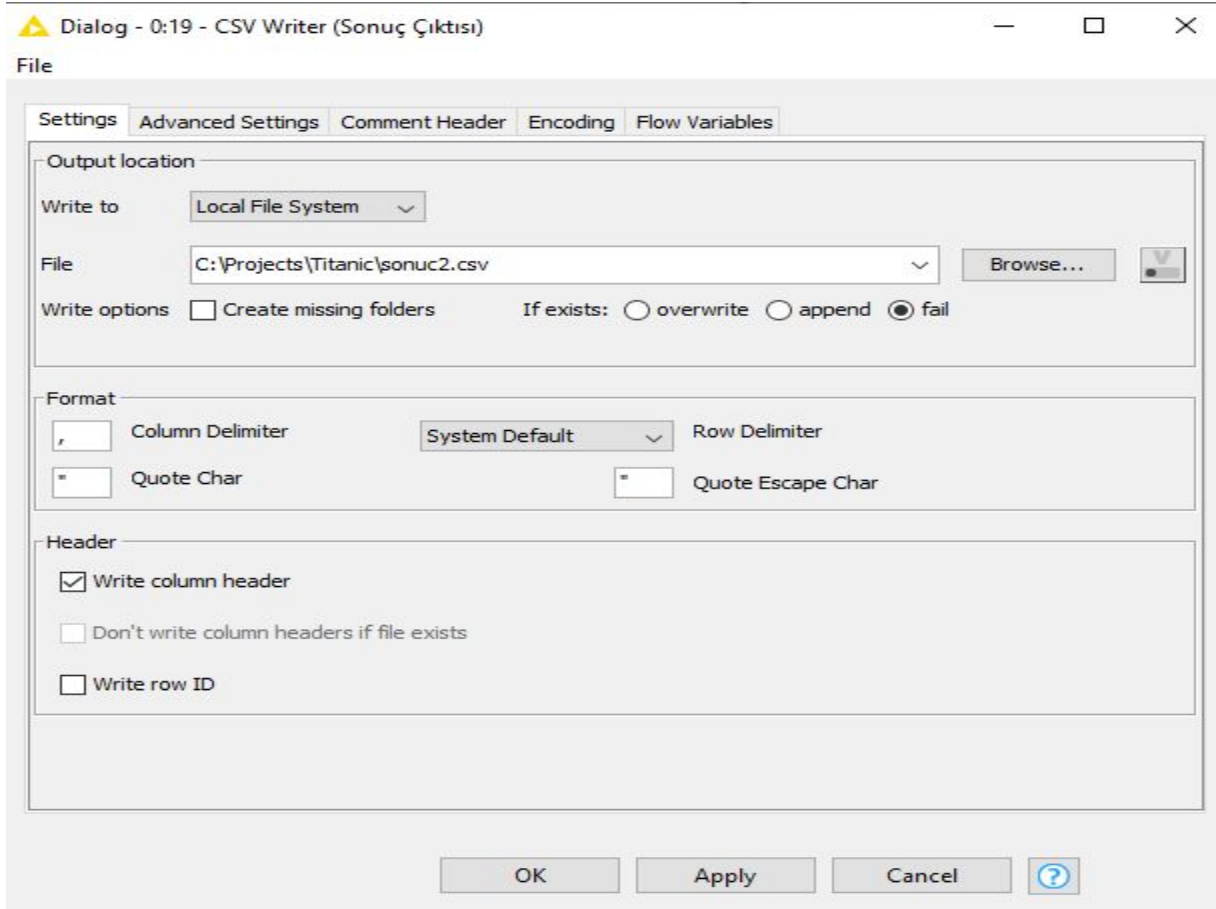
☐ Enforce inclusion

Hayatta Kalma Kategorizeleştirmesini sonuçta göstermek için geri alıyoruz ve yine 1,0 gösterim şeklini alıyor.

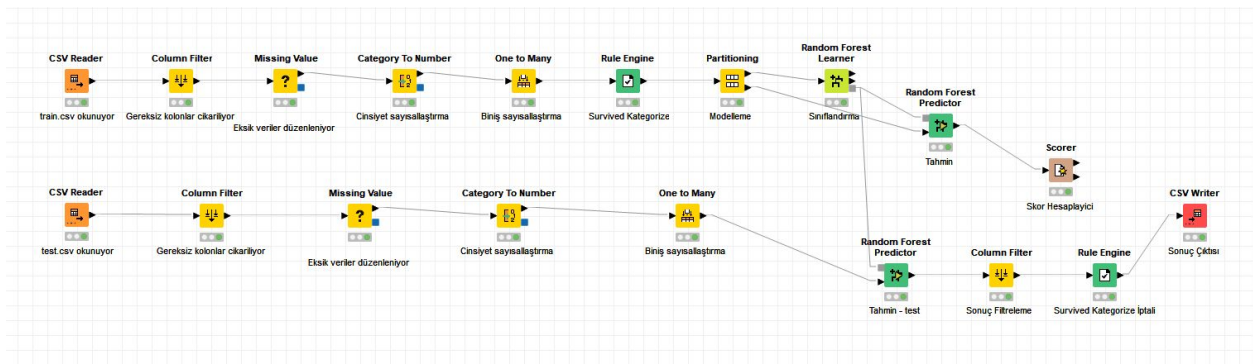


Sonucu Dosyaya Yazdırma

Elde ettiğimiz sonucu Kaggle platformuna submission olarak göndermek için CSV'ye yazdırmamız gerekiyor. Bunun için son olarak CSV Writer node'u kullanıyoruz.



Son olarak KNIME platformunda yaptıklarımız aşağıdaki gibi görünmektedir:



Yöntemleri Karşılaştırma

İki yöntemden de elde ettiğimiz sonuç csv dosyalarını Kaggle platformunda Submission olarak ekledik ve aşağıdaki sonuçları elde ettik:

Submission and Description	Public Score
sonuc.csv 12 minutes ago by Onur Osman Güle Knime ile yapılan G171210021 ve G171210375 çalışması.	0.77751
notebook1a2720b400 SAU Son (version 1/1) 20 minutes ago by Onur Osman Güle From "notebook1a2720b400" Notebook	0.77272

Sonuç

Python ve KNIME kullanarak Titanic'teki yolcu listesini işleyip kimlerin kurtulduğunu tahmin etmeye çalıştık. Tahminimiz %77 oranda doğru oldu.

KNIME uygulaması, python'dan çok daha kolay ve kod yazmadan kullanıldığından yarışmayı oradan çözmek daha rahattı.

Farklı algoritmalar kullanılarak model geliştirirsek bulduğumuz %77 doğruluk oranını daha da yükseğe çekebiliriz.

Bağlantılar

Proje Github: <https://github.com/onurgule/Big-Data-Titanic/>

Kaggle Notebook: <https://www.kaggle.com/onurgule/notebook1a2720b400>