T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Büyük Veriye Giriş Dersi Proje Ödevi

Titanic - Machine Learning from Disaster

B191210004 – Mustafa Melih Tüfekcioğlu – 1.Öğretim A Grubu
mustafa.tufekcioglu@ogr.sakarya.edu.tr

B181210010 - Deniz Berfin Taştan – 1.Öğretim B Grubu
deniz.tastan@ogr.sakarya.edu.tr

SAKARYA

Aralık 2022

Büyük Veriye Giriş Dersi

1.Proje Amacı

Proje Titanic veri seti kullanarak bir model eğitmek üzerine yapıldı. Proje Python dili ile yazıldı. Eğitilen model gemi enkazından hangi yolcuların sağ çıktığını tahmin etmek için kullanılacak. Modeli eğitmek için izlenecek adımlar şu şekilde:

1. Kullanılacak olan veri setini analiz etme
2. Veri setindeki değişkenlerin birbirine gör bağımlılıklarını hesaplama
3. Tespit edilen bağımlılıklara göre kullanılacak olan değişkenleri tespit etme
4. Model eğitimi için kullanılacak veri setini eğitime uygun hale getirme (boş alan doldurma, değişken birleştirme vs.)
5. Final veri seti oluşturma
6. Model eğitme ve çıkan doğruluk değerlerine göre algoritma seçimi
7. Sonuçlandırma

Proje dosyasında 2 adet .py uzantılı dosya bulunuyor. Titanic.py dosyası veri hazırlama, eğitim ve test kodlarını içerirken, plot.py dosyası verileri analiz ettikten sonra grafik çizme kodlarını içeriyor.

2. Grup Üyelerinin Projeye Katkısı:

Projede veri setinin analiz edilmesi, plot kütüphanesini kullanarak grafik çizimleri (plot.py) ve raporlama işlemlerini Deniz Berfin Taştan yapmıştır.

Makine öğrenmesi algoritmaları ile sonuçlandırma (titanic.py) aynı zamanda bu yazılan kodların birleştirilmesi (titanic.ipyn) işlemlerini Mustafa Melih Tüfekcioğlu yapmıştır.

3. Projede Kullanılacak Veri Seti Hakkında

3.1 Eğitim için kullanılacak veri seti (train.csv)

Modeli eğitmek için kullanılacak olan Train.csv dosyası 892 tane veriden oluşuyor. Her veri bir kişiyi temsil etmekte ve dosyada bir kişinin aşağıdaki bulunmaktadır.

-PassengerID (Her kişi için verilen bir ID numarası)

-Survived (Kişinin kaza esnasında hayatta kalıp kalmadığı hakkında bilgi)

-PClass (Kişinin hangi bilet sınıfına sahip olduğu bilgisi)

-Name (Kişinin adı ve soyadı)

-Sex (Kişinin cinsiyeti)

-Age (Kişinin yaşı)

-SibSp (Kişinin eş/kardeş sayısı)

-Parch (Kişinin ebeveyn/çocuk sayısı)

-Ticket (Kişinin bilet bilgisi)

-Fare (Kişinin ödediği ücret)

-Cabin (Kişinin kabin bilgisi)

-Embarked (Kişinin gemiye nereden bindiğinin bilgisi)

Veri seti içerisinde tekrarlanan bir veri bulunmuyor fakat bazı sütunlarda boş alanlar bulunuyor. Verileri matematiksel olarak sınıflamak istediğimizde de aşağıdaki tabloyu elde ediyoruz.

```
train.describe().T
```

	count	mean	std	min	25%	50%	75%	max
PassengerId	891.0	446.000000	257.353842	1.00	223.5000	446.0000	668.5	891.0000
Survived	891.0	0.383838	0.486592	0.00	0.0000	0.0000	1.0	1.0000
Pclass	891.0	2.308642	0.836071	1.00	2.0000	3.0000	3.0	3.0000
Age	714.0	29.699118	14.526497	0.42	20.1250	28.0000	38.0	80.0000
SibSp	891.0	0.523008	1.102743	0.00	0.0000	0.0000	1.0	8.0000
Parch	891.0	0.381594	0.806057	0.00	0.0000	0.0000	0.0	6.0000
Fare	891.0	32.204208	49.693429	0.00	7.9104	14.4542	31.0	512.3292

3.2 Test için kullanılacak veri seti (test.csv)

Test için kullanılacak veri seti 419 tane veriden oluşuyor. Test veri seti, eğitim veri seti içindeki tüm sütunlara sahip fakat test veri setinde Survived sütunu bulunmuyor. Veri seti içerisinde tekrarlanan bir veri bulunmuyor fakat bazı sütunlarda boş alanlar bulunuyor.

4. Veri Setindeki Bilgilerin Birbirine Göre Bağımlılıkları

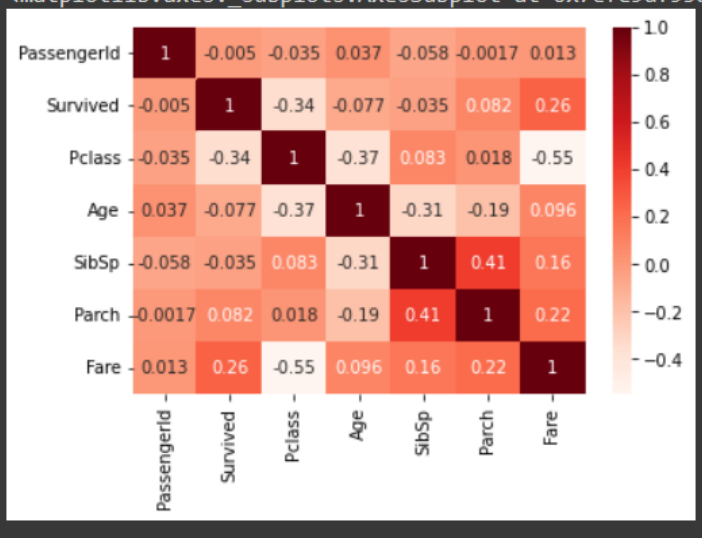
Veri setindeki değişkenlerin birbirlerine göre bağımlılıklarını ölçmek için korelasyon katsayısı hesaplaması yapılır. Bu katsayı -1 ile +1 arasında bir değer alır. Katsayı +1'e veya -1'e ne kadar yakınsa ilişkinin bağımlılığı o kadar güçlüdür. Korelasyon katsayısı aşağıdaki formül ile hesaplanır.

$$\rho_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}\,\sqrt{E(Y^2) - E^2(Y)}}$$

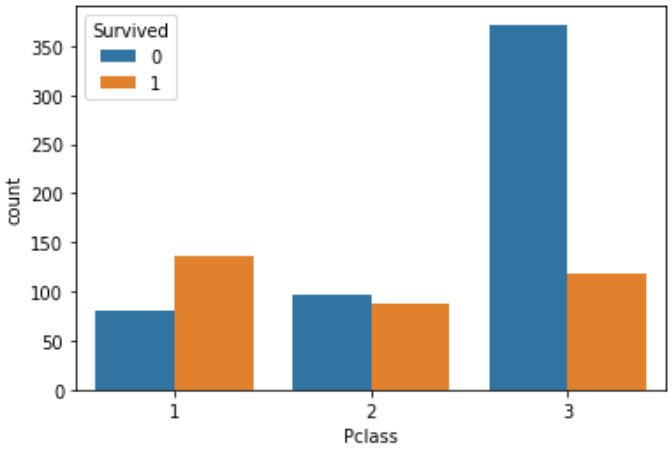
Python'da korelasyon katsayını hesaplamak için seaborn kütüphanesinin corr() fonksiyonu kullanılır. Veri setimiz için bu fonksiyonu kullandığımızda aşağıdaki tabloyu elde ediyoruz.

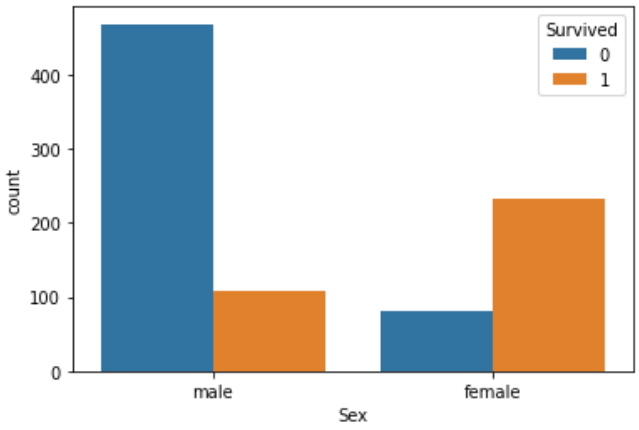
```
train.corr()
sns.heatmap(train.corr(),cmap="Reds", annot=True)
```

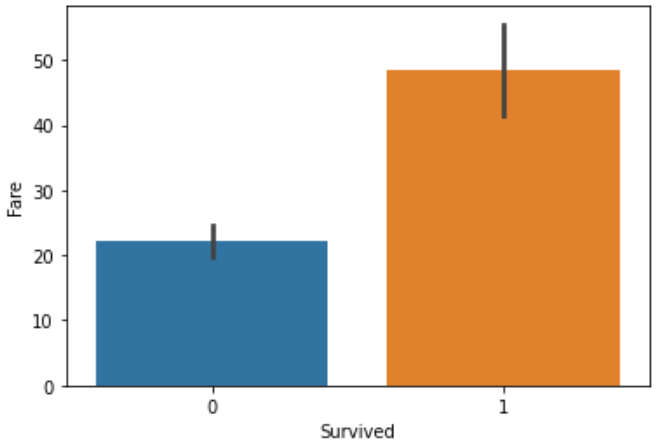
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efe9df95810>
```

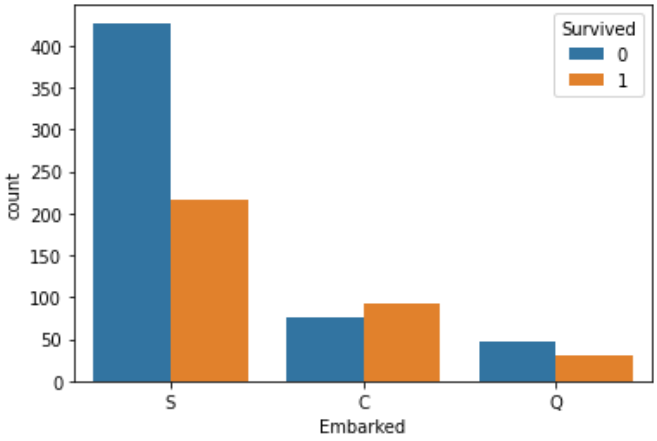


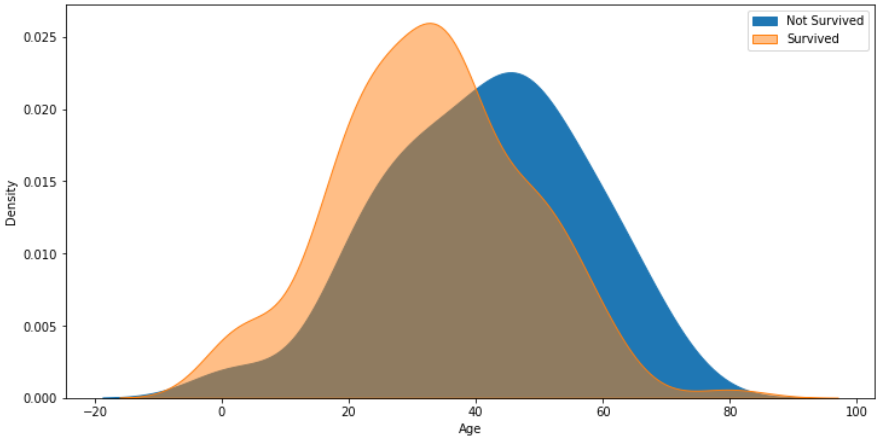
Hayatta kalma bilgisinin diğer değişkenlerle olan grafikleri:











5.Veri Düzenleme

Veri setlerini projemizde kullanmamız için önce düzenlememiz gerekiyor. Önce veri setleri içerisindeki boş alanları doldurma işlemi yaptık.

Train.csv içerisindeki boş alanları bulmak için **".isnull().sum()"** fonksiyonunu kullandık. Çıkan sonuca göre *yaş(age), konum (embarked) ve kabin (cabin)* alanlarında boşluklar bulunuyor.

- *Yaş (age)* bilgisindeki boşlukları doldurmak için ilgili veri setindeki yaşların ortalamasını hesapladık ve çıkan sonucu boş alanlara eklendi.
- *Kişinin gemiye bindiği konum bilgisindeki (Embarked)* boş alanları doldurmak için ilgili sütundaki en çok tekrar eden veri olan "S" değeri boş alanlara eklendi.
- *Kabin (Cabin) bilgisinin* elde etmek istediğimiz hayatta kalma (survived) bilgisini tahmin etmede etkisi olmadığı için veri setinden silindi.

Test.csv içerisindeki boş alanları bulmak için tekrar aynı fonksiyonu kullandık. Çıkan sonuca göre yaş(age), ücret(fare) ve kabin(cabin) alanlarında boşluklar bulunuyor.

- *Yaş (age)* bilgisindeki boşlukları doldurmak için ilgili veri setindeki yaşların ortalamasını hesapladık ve çıkan sonucu boş alanlara eklendi.
- *Ücret (fare)* bilgisindeki boşlukları doldurmak için ilgili veri setindeki ücretlerin ortalamasını hesapladık ve çıkan sonucu boş alanlara eklendi.
- *Kabin (Cabin) bilgisinin* elde etmek istediğimiz hayatta kalma (survived) bilgisini tahmin etmede etkisi olmadığı için veri setinden silindi.

```
train.isnull().sum()
train['Age'].fillna(train['Age'].mean(),inplace=True)
train['Embarked'].fillna('S',inplace=True)
train.drop(columns=['Cabin'],inplace=True)

test.isnull().sum()
test.drop(columns=['Cabin'],inplace=True)
test['Fare'].fillna(test['Fare'].mean(),inplace=True)
test['Age'].fillna(test['Age'].mean(),inplace=True)
```

Boş alanları doldurduktan sonra veri setlerindeki ihtiyaç olmayan verileri silmemiz gerekiyor. Örneğin *isim (name)* ve *bilet (ticket)* bilgilerinin *kişinin hayatta kalma (survived)* bilgisine bir etkisi olmadığı için ilgili sütunlar silindi. Daha sonra ilgili verileri birleştime işlemi yapıldı. *Kardeş/eş sayısı (Sibsp) ve ebeveyn/çocuk (Parch)* sayısının toplamı *temp_family* geçici değişkenine atandı ve bu sütunlar veri setinden silindi.

```
[73] train.drop(columns=['Name','Ticket'],inplace=True)
     test.drop(columns=['Name','Ticket'],inplace=True)
     train['temp_Family']=train['SibSp'] + train['Parch']
     test['temp_Family']=test['SibSp'] + test['Parch']
     train.drop(columns=['SibSp','Parch'],inplace=True)
     test.drop(columns=['SibSp','Parch'],inplace=True)
```

Birleştirme işleminden sonra veri setlerine *Family* sütunu eklendi ve kişiler aile sayısına göre etiketlendi. Daha sonra veri setlerine *AgeGroup* sütunu eklendi ve kişiler yaş gruplarına göre etiketlendi.

```
ages= [0,16,30,45,50]
labels = ['Children','Young','Adult','Old']
train['AgeGroup'] = pd.cut(train['Age'], bins=ages, labels=labels, right=False)
test['AgeGroup'] = pd.cut(train['Age'], bins=ages, labels=labels, right=False)
```

```
[75] persons= [0,1,5,7,11]
     labels = ['Alone','Dou','Small','Large']
     train['Family'] = pd.cut(train['temp_Family'], bins=persons, labels=labels, right=False)
     test['Family'] = pd.cut(train['temp_Family'], bins=persons, labels=labels, right=False)
```

6. Kullanılan Büyük Veri Platformları

Projede büyük veri platformu kullanılmamıştır. Veri işlemek için Python'un kütüphaneleri (pandas, numpy, seaborn, matplotlib, sklearn) kullanılmıştır.

7. Final Veri Seti Oluşturma ve Model Eğitimi

Veri düzenleme işlemlerinden sonra model eğitimi ve sonuç tahmin işlemleri için kullanılacak olan final veri seti oluşturuldu. Bu veri setine eğitimde kullanılacak olan *Pclass, Sex, Embarked, AgeGroup, Family* bilgileri eklendi

```
final_train=pd.get_dummies(train, columns=['Pclass','Sex','Embarked','AgeGroup','Family'], drop_first=True)
final_test=pd.get_dummies(test, columns=['Pclass','Sex','Embarked','AgeGroup','Family'], drop_first=True)
```

```
x_test=final_train.drop(columns="Survived")
y_test=final_train['Survived']
x_train, x_test, y_train, y_test=train_test_split(x_test,y_test,test_size=0.2,random_state=42)
y_train
```

Tanımlanan **x_test** değişkeni bağımsız değişkenleri temsil ettiği için tahmin edilmesi istenen ve bağımlı değişken olan *Survived* sütununu veri setinden siliyoruz. **Y_test** ise bağımlı yani tahmin edilecek olan değişkeni ifade ediyor. Model eğitiminde kullanılacak olan verileri test ve train olarak bölmek için de **Sklearn** kütüphanesinin **train_test_split** fonksiyonunu kullanıyoruz. Bu fonksiyonun içerisine yazılan **test_size** parametresini 0.2 vererek final veri setinin %80'ininı eğitim için, %20'sini de test için kullanıyoruz.

Eğitilecek tüm verileri ölçeklendirme için MinMaxScaler fonksiyonunu kullandık. Bu sayede tüm verilerimiz sayısal değere çevrilmiş oldu.

final_train - DataFrame

Index	PassengerId	Survived	Age	Fare	emp.Family	Pclass 2	Pclass 3	Sex_male	Embarked_Q	Embarked_S	AgeGroup_Young Adults	AgeGroup_Middle-aged Adult	AgeGroup_Old Adults	Family_Dou	Family_Small	Family_Large
0	1	0	22	7.25	1	0	1	1	0	1	1	0	0	1	0	0
1	2	1	38	71.2833	1	0	0	0	0	0	0	1	0	1	0	0
2	3	1	26	7.925	0	0	1	0	0	1	1	0	0	0	0	0
3	4	1	35	53.1	1	0	0	0	0	1	0	1	0	1	0	0
4	5	0	35	8.05	0	0	1	1	0	1	0	1	0	0	0	0
5	6	0	29.6991	8.4583	0	0	1	1	1	0	1	0	0	0	0	0
6	7	0	54	51.8625	0	0	0	1	0	1	0	0	0	0	0	0
7	8	0	2	21.075	4	0	1	1	0	1	0	0	0	1	0	0
8	9	1	27	11.1333	2	0	1	0	0	1	1	0	0	1	0	0
9	10	1	14	30.0708	1	1	0	0	0	0	0	0	0	1	0	0
10	11	1	4	16.7	2	0	1	0	0	1	0	0	0	1	0	0
11	12	1	58	26.55	0	0	0	0	0	1	0	0	0	0	0	0
12	13	0	20	8.05	0	0	1	1	0	1	1	0	0	0	0	0

Model eğitimi için makine öğrenmesi algoritmalarından Logistic Regression, Decision Tree, K-Neighbors, Random Forest, Support Vector Machine ve Naive Bayes algoritmalarını kullandık.

```
lr=LogisticRegression(random_state = 72)
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)

clf=DecisionTreeClassifier(random_state = 72)
clf.fit(x_train,y_train)
y_pred_D=clf.predict(x_test)

knn_classifier = KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(x_train, y_train)
y_pred_knn = knn_classifier.predict(x_test)

rm_classifier = RandomForestClassifier()
rm_classifier.fit(x_train,y_train)
y_pred_rm=rm_classifier.predict(x_test)

gnb_classifier = GaussianNB()
gnb_classifier.fit(x_train,y_train)
y_pred_gnb = gnb_classifier.predict(x_test)

svm_classifier = SVC(kernel='linear')
svm_classifier.fit(x_train, y_train)
y_pred_svm = svm_classifier.predict(x_test)
```

8. Proje Çıktıları

```
Accuracy Scores:
SVM: 0.7877094972067039
GNB:  0.441340782122905
Random Forest: 0.8435754189944135
Lojistik Regresyon: 0.8100558659217877
Karar Agaclari: 0.770949720670391
KNN: 0.8268156424581006
```

Eğitimden sonra test verileri ile doğrulama yaptığımız zaman en yüksek doğruluk sonucunu veren algoritma olduğundan dolayı sonuçlara erişmek için seçtiğimiz algoritma Random Forest Algoritması oldu. Bu algoritmayı kullandıktan sonra, içerisinde tahmin sonuçları ve passenger ID'yi barından bir DataFrame oluşturduk. Daha sonra oluşturduğumuz bu DataFrame'i CSV dosyası formatında kaydettik.

	A	B	C	D	E
1	PassengerId	Survived			
2	892				
3	893				
4	894				
5	895				
6	896				
7	897				
8	898	1			
9	899				
10	900	1			
11	901				
12	902	1			
13	903	1			
14	904	1			
15	905				
16	906	1			
17	907	1			
18	908				
19	909				
20	910				
21	911	1			
22	912				
23	913				
24	914	1			
25	915				
26	916	1			

Hayatta kalan yolcuların Survived değeri 1 olarak belirleyip dosyaya bu şekilde yazdık.