

**TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**MAKİNE ÖĞRENMESİ SINIFLANDIRMA
ALGORİTMALARI KULLANILARAK TRAFİK
KAZALARININ ŞİDDETİNİN TAHMİNİ**

17011090 – Tarık Can ŞAHİN
16011036 – İlker BEDİR

BİLGİSAYAR PROJESİ

Danışman
Öğr.Gör. Furkan ÇAKMAK

Ocak, 2021

TEŞEKKÜR

Bu projenin gerçekleştirilmesi aşamasında değerli bilgilerini bizimle paylaşan, kendisine daniştiğimiz birçok konuda değerli desteklerini bizden esirgemeyen, güler yüzünü ve samimiyetini her daim gösteren, bizleri araştırmaya ve öğrenmeye, çeşitli kavramları bilgi dağarcığımıza katmamıza olanak sağlayan, karşılaşlığımız sorunlara efektif ve etkili çözümler getiren, kıymetli ve danışman hoca statüsünü hakkıyla yerine getiren Öğr.Gör. Furkan ÇAKMAK'a teşekkürü borç bilir ve şükranlarını sunarız.

Eğitim hayatımızda önemli rolleri olan saygı değer diğer üniversite hocalarımızın da bizlere 3 yıldan fazladır devam eden üniversite hayatımız boyunca kazandırdıkları her değerli bilgi için ve bizi gelecekte nitelikli ve değerli kılacak bilgilerle donattıkları için her birine teker teker teşekkürlerimizi sunarız. Son olarak çalışmamızda desteğini ve bize olan güvenini bizden esirgemeyen Yunus KARATEPE'ye ve bizleri bu günlere büyük bir özen ve çaba göstererek, saygı ve sevgi kavramlarını benimseterek getiren ve bizden hiçbir zaman desteğini esirgemeyen bu hayattaki en büyük şansımız olan ailelerimize sonsuz teşekkürler.

Tarık Can ŞAHİN
İlker BEDİR

İÇİNDEKİLER

KISALTMA LİSTESİ	v
ŞEKİL LİSTESİ	vi
TABLO LİSTESİ	ix
ÖZET	x
ABSTRACT	xii
1 GİRİŞ	1
2 ÖN İNCELEME	2
2.1 LİTERATÜR ARAŞTIRMASI	2
3 FİZİBİLİTE	9
3.1 Genel Gereksinim Karşılaştırma Tabloları	9
3.2 Teknik Fizibilite	10
3.2.1 Yazılım Fizibilitesi	10
3.2.2 Donanım Fizibilitesi	10
3.3 İş gücü ve Zaman Planlaması	11
3.3.1 Zaman Fizibilitesi Ve Gantt Diyagramı	11
3.4 Yasal Fizibilite	11
3.5 Ekonomik Fizibilite	11
4 Sistem Analizi	12
4.1 Kullanılan Araştırma ve Bilgi Toplama Yöntemleri	12
4.2 Sistem Modülleri	12
4.2.1 DTA Model Modülü	12
4.2.2 ASM (Attribute Selection Measures Nedir?)	13
4.2.3 RFA Model Modülü	15
4.2.4 NBA Model Modülü	17
4.2.5 Gaussian NBA Model Modülü	19
4.2.6 Bernoulli NBA Model Modülü	20

4.2.7 Multinomial NBA Model Modülü	21
5 Sistem Tasarımı	23
5.0.1 İş Akış Diyagramı	23
5.0.2 Context Diyagramı	24
5.0.3 Alt Seviye Veri Akış Diyagramı	24
5.0.4 0.Düzey Veri Akış Diyagramı	25
6 Uygulama	26
6.1 Veri Setinin Düzenlenmesi	26
7 Performans Analizi	34
7.1 Veri Setinde Yapılan Güncellemeler	40
7.1.1 Ortalama Değerler	40
7.1.2 Regression Yöntemi	41
7.1.3 MICE Modeli ve Regression Yöntemi Entegresi	42
8 Sonuç	44
Referanslar	46
Özgeçmiş	47

KISALTMA LİSTESİ

AS	Accuracy Score
ASM	Attribute Selection Measures
CM	Confusion Matrix
DTA	Decision Tree Algorithm
MICE	Multiple Imputation by Chained Equations
ML	Machine Learning
NBA	Naive-Bayes Algorithm
RFA	Random Forest Algorithm
USA	United States of America

ŞEKİL LİSTESİ

Şekil 2.1 Logistic Regression Doğruluk Oranı	3
Şekil 2.2 K-Nearest Neighbors Algoritması Doğruluk Oranı	4
Şekil 2.3 Optimize Edilmiş K-Nearest Neighbors Algoritması	4
Şekil 2.4 Decision Tree Algoritması Doğruluk Oranı	5
Şekil 2.5 Random Forest Algoritması Doğruluk Oranı	5
Şekil 2.6 Severity Değerini En Çok Etkileyen 10 Faktör	6
Şekil 2.7 Severity Değerini En Çok Etkileyen k Faktör Sıralaması	6
Şekil 2.8 Severity Değerini En Çok Etkileyen k Faktöre Göre Verinin Düzenlenmesi ve Train-Test Ayrımı	7
Şekil 2.9 Severity Değerini En Çok Etkileyen k Faktöre Göre Modelin Random Forest Algoritması ile Eğitilmesi	8
Şekil 2.10 4 Algoritmanın Karşılaştırılması	8
Şekil 3.1 Gantt Diyagramı	11
Şekil 4.1 DTA Sınıflandırma Mantığı	13
Şekil 4.2 Information Gain Hesabı	14
Şekil 4.3 Split Info ve Gain Ratio değerinin hesaplanması	14
Şekil 4.4 Gini hesabı	15
Şekil 4.5 RFA Çalışma Mantığı	16
Şekil 4.6 Class Conditional Independence Olasılık hesabı	17
Şekil 4.7 Likelihood tabloları oluşturma ve Naive Bayes olasılığının hesaplanması	18
Şekil 4.8 Gaussian NBA Olasılık Hesabı	19
Şekil 4.9 Gaussian NBA Sınıflandırma	19
Şekil 4.10 Bernoulli NBA Olasılık Hesabı	20
Şekil 4.11 Bernoulli NBA Sınıflandırma	20
Şekil 4.12 MultiNomial NBA Normal Mail Örnek	21
Şekil 4.13 MultiNomial NBA Spam Mail Örnek	22
Şekil 5.1 İş akış diyagramı	23
Şekil 5.2 Context Diyagramı	24
Şekil 5.3 Alt Düzey Veri Akış Diyagramı	24
Şekil 5.4 0. Düzey Veri Akış Diyagramı	25

Şekil 6.1 Korelasyon Haritası	27
Şekil 6.2 Eyaletlerdeki Kaza Sayıları	28
Şekil 6.3 Kaza Siddeti Çeşitleri ve Sayıları	29
Şekil 6.4 Kazanın En Çok Hangi Hava Koşulunda Gerçekleştiğini Belirten Grafik	29
Şekil 6.5 Random Forest % 20 Test Boyutuna Göre CM	30
Şekil 6.6 Decision Tree % 20 Test Boyutuna Göre CM	30
Şekil 6.7 Gaussian Naive Bayes % 20 Test Boyutuna Göre CM	31
Şekil 6.8 Algoritmaların Doğruluk Değerleri	31
Şekil 6.9 Algoritmaların Çalışma Süreleri	32
Şekil 6.10 Algoritmaların F-1 Score Değerleri	32
Şekil 6.11 Algoritmaların Precision Değerleri	32
Şekil 6.12 Algoritmaların Recall Değerleri	33
Şekil 7.1 Test boyutu = 0.1 algoritmaların karmaşıklık matrisleri (Confusion Matrix)	34
Şekil 7.2 Test boyutu = 0.1 algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri ve hız verisi	35
Şekil 7.3 Test boyutu = 0.2 algoritmaların karmaşıklık matrisleri (Confusion Matrix)	36
Şekil 7.4 Test boyutu = 0.2 algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri ve hız verisi	36
Şekil 7.5 Test boyutu = 0.3 algoritmaların karmaşıklık matrisleri (Confusion Matrix)	37
Şekil 7.6 Test boyutu = 0.3 algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri ve hız verisi	37
Şekil 7.7 Test boyutu = 0.5 algoritmaların karmaşıklık matrisleri (Confusion Matrix)	38
Şekil 7.8 Test boyutu = 0.5 algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri ve hız verisi	38
Şekil 7.9 Test boyutu = 0.7 algoritmaların karmaşıklık matrisleri (Confusion Matrix)	39
Şekil 7.10 Test boyutu = 0.7 algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri ve hız verisi	39
Şekil 7.11 Max Depth parametresinin incelenmesi	39
Şekil 7.12 Test boyutu = 0.2 DTA Karmaşıklık Matrisi	40
Şekil 7.13 Test boyutu = 0.1 RFA Karmaşıklık Matrisi	41
Şekil 7.14 Test boyutu = 0.2 Regression Yöntemiyle doldurulan veriler ve MICE modeli entegresi sonucu oluşan her bir algoritmanın karmaşıklık matrisleri	42

TABLO LİSTESİ

Tablo 3.1 İşlemci	9
Tablo 3.2 İşletim Sistemi	9
Tablo 3.3 RAM	9
Tablo 3.4 Yazılım Dili	10
Tablo 3.5 Algoritma	10
Tablo 3.6 Geliştirme Ortamı	10
Tablo 4.1 RFA Avantaj ve Dezavantajları	16
Tablo 4.2 NBA Avantaj ve Dezavantajları	18
Tablo 4.3 Bernoulli NBA Avantaj ve Dezavantajları	20

ÖZET

MAKİNE ÖĞRENMESİ SINIFLANDIRMA ALGORİTMALARI KULLANILARAK TRAFİK KAZALARININ ŞİDDETİNİN TAHMİNİ

Tarık Can ŞAHİN

İlker BEDİR

Bilgisayar Mühendisliği Bölümü

Bilgisayar Projesi

Danışman: Öğr.Gör. Furkan ÇAKMAK

Günümüzde araç sayılarının da artışı ile beraber neredeyse her gün trafik kazaları meydana gelebilmektedir. Bu kazalar öncelikli olarak kaza yapan araç sahiplerini veya kazazedeleri hem maddi hem de ruhsal yönden oldukça etkileyebilmektedir. Ayrıca trafik kazaları, kazadan doğrudan etkilenmeyen sürücülerin günlük aktivitelerinin gerçekleştirilemesinde veya başka bir yere ulaşımlarında zaman bakımından aksamalara yol açabilmektedir. Bu proje ile bir bölgede meydana gelen trafik kazasının trafiğe olan etkisi incelenecik olup o bölgede herhangi bir kaza gerçekleşmesi durumunda trafik akışına etkisi derece olarak tahmin edilebilecektir.

Proje için fonksiyonel bir dil olduğundan "Python" yazılım dilinin kullanılmıştır. Grafikler ve istatistiksel veriler için PYTHON'ın "matplotlib" ve "numpy" kütüphaneleri aracılığıyla birçok istatistiksel tahmin verileri incelenmiş ve en doğru tahmin için en uygun makine öğrenmesi algoritması karşılaştırmalarla birlikte detaylıca anlatılmıştır. Projenin gerçekleştirme aşamasında algoritmanın çalışma süresinin hesaplanması, kullanıcıya bildirilmesi ve en hızlı çözüm üretebilen algoritmanın belirlenmesi için belirli satırda başlayarak mevcut saat alabilmesi için "datetime" kütüphanesi kullanılmıştır.

Makine öğrenmesi tekniklerinden Random Forest, Decision Tree ve Naive Bayes algoritmaları için "scikit-learn" kütüphanesi kullanılmış ve her bir algoritma için "tree", "RandomForestClassifier" ve "naive-bayes" kütüphaneleri, bu algoritmaların accuracy

score değerlerinin elde edilebilmesi için "accuracy-score" kütüphanesi kullanılmıştır. Ayrıca her bir algoritmanın confusion matrix değerleri incelenmiş olup ilerleyen bölümlerde sonuçlar detaylıca anlatılmıştır. Veri setinde eksik değerlere sahip satırlar kaldırılmış olup, tahmin için etkili olan kolonlardan sıcaklık kolonunun bazı değerlerinin negatif olmasından dolayı sıcaklığın Kelvin biriminde olacak şekilde düzenlenmesi kararlaştırılmıştır.

Makine öğrenmesi metotları olarak güvenilir ve büyük verilerde yüksek doğruluk payı içeren ve hızlı çalışan bir algoritma olduğundan Naive Bayes kullanılmıştır. Bölgede gerçekleşen kazanın trafik akışına olan olumsuz etkisinin "severity" verisine göre sınıflandırılarak öğrenilmesi için Decision Tree ve Random Forest algoritmalarının kullanılmıştır. Her bir algoritmanın optimize edilip en yüksek doğruluk payına sahip sonuçlarının elde edilebilmesi için algoritmaların çağırılan fonksiyonlarının kendi içlerinde sahip oldukları parametrelerin değerleri sürekli değiştirilmiş ve elde edilen doğruluk değerleri, bu değişen parametre değerlerine göre grafikler halinde gösterilmiştir. Son olarak elde edilen istatistiksel veriler incelenmiş ve en efektif algoritma belirlenmiştir.

Anahtar Kelimeler: Trafik Kazaları, şiddet, tahmin, rastgele orman, makine öğrenmesi, naive bayes, karar ağacı, sınıflandırma

ABSTRACT

PREDICTION OF SEVERITY OF TRAFFIC ACCIDENTS BY USING MACHINE LEARNING CLASSIFICATION ALGORITHMS

Tarık Can ŞAHİN
İlker BEDİR

Department of Computer Engineering
Computer Project

Advisor: Lect. Furkan ÇAKMAK

Nowadays, with the increase in the number of vehicles, traffic accidents can occur almost every day. These accidents can primarily affect the vehicle owners or victims of the accident both financially and mentally. In addition, traffic accidents can cause disruptions in terms of time in the daily activities of the drivers who are not directly affected by the accident or in their reach to another place. With this project, the effect of a traffic accident on traffic in a region will be analyzed, and in the event of an accident in that region, its effect on traffic flow can be predicted in degrees.

Since it is a functional language for the project, "Python" programming language was used. For graphs and statistical data, Python's "matplotlib" and "numpy" libraries were used and many statistical prediction data were examined. According to these examinations the most appropriate machine learning algorithm for the most accurate prediction was explained by detailed comparisons. In the realization phase of the project, "datetime" library has been used to calculate the working time of the algorithm also to determine the algorithm that can produce the fastest solution and inform the user.

For "Random Forest Classifier", "Decision Tree" and "Naive Bayes" machine learning algorithms, "scikit-learn" library had been used. "tree", "RandomForestClassifier" and "naive-bayes" libraries were used for basic machine learning operations. Also by using "accuracy-score" library, accuracy score had been calculated of each algorithm.

Confusion matrix values of each algorithm had been examined and the investigations are explained in detail in the following sections. The rows with missing values in the data set have been removed. Also due to the negative values of the temperature column, one of the columns effective for estimation, it has been decided to arrange the temperature in Kelvin unit.

Naive Bayes has been used as machine learning methods because it is a reliable and fast-running algorithm with high accuracy in big data. Decision Tree and Random Forest algorithms were used to learn the negative impact of the accident on the traffic flow in the region by classifying them according to "severity" data. In order to optimize each algorithm and obtain the results with the highest accuracy, the values of the parameters that the algorithms have called functions are constantly changed, and the obtained accuracy values are shown in graphs according to these changing parameter values. Finally, the statistical data obtained were analyzed and the most effective algorithm was determined.

Keywords: Traffic Accidents, severity, prediction, random forest, machine learning, naive bayes, decision tree, classification

1 GİRİŞ

Tüm dünyadaki yenilik ve gelişmeler yük ve yolcu hareketliliğini de doğrudan etkilemektedir. İnsanların ekonomik durumunun iyileşmesi, artan ticaret hacmi, malların dolaşımının artması ve hızlanması, yol ve araç kalitesinin iyileşmesi, ulaşım modlarının yön değiştirmesi ve insanları yeni alternatifler sunması gibi etkenler bir şekilde ulaşım sistemlerini ve güvenliğini de etkilemektedir. Trafiğe kayıtlı araç sayısının artması, nüfus artışı, sürücü belgesine sahip kişi sayısının artması, taşımacılık sektöründe hız ve rekabetin öne çıkması, insanların önceki yillarda göre daha mobil bir iş ve yaşam tarzına yönelmeleri gibi nedenlerinde etkisi ile trafik yoğunluğu artmakta ve bu durum trafik güvenliğine de yansımaktadır. Dünya Sağlık Örgütü rakamlarına göre ne yazık ki dünyada her yıl bir milyondan fazla insan trafik kazası sonucu hayatını kaybetmektedir. Dünyada insanların ölüm nedenlerine bakıldığında da trafik kazası nedeni ile ölümler her yıl üst sıralara yükselmeye devam etmektedir. 2030 yılına gelindiğinde ölüm nedenleri arasında trafik kazası sonucu ölümlerin beşinci sıraya yükseleceği öngörmektedir. Ayrıca trafik kazası sonucu ölenler arasında genç oranı da oldukça yüksektir. Ölümler yanında yaralanmalar ve sakat kalan insanlar ile uğranılan ekonomik kayıplar da düşünüldüğünde trafik kazalarını önlemeye yönelik çalışmaların ne kadar önemli ve vazgeçilmez olduğu gerçeği ortaya çıkmaktadır.

Bu proje, kayda geçmiş olan trafik kazalarının kaza sırasında hava durumu, yol durum bilgisi parametrelerini de göz önüne alarak şiddetini tahmin etmek amacıyla gerçekleştirılmıştır. Bu proje ile gelecekte kaza etkisini azaltmak için alınması gereken önlemlerin kararlaştırılmasının önünü açacağımı düşündürüyoruz.

Projemiz bir eğitim öğretim çalışması olması sebebiyle öğretmeyi ve öğrenmeyi amaçlamaktadır. Bu projedeki destekleri için danışman hocamıza ve yakın arkadaşlarımıza teşekkür ediyoruz.

Bu çalışmanın tüm takipçilerine faydalı olması dileğiyle...

2 ÖN İNCELEME

2.1 LİTERATÜR ARAŞTIRMASI

Bu kısımda projede kullanılacak algoritma olarak seçilen Decision Tree Algorithm (Karar Ağacı Algoritması) ile veri kümesi olarak seçilen Trafik kazaları veri seti ile işleyişin araştırmasının sonuçları yer almaktadır.

Kaggle, makine öğrenmesi ile uğraşan birçok yazılım geliştiricisinin veri seti ihtiyacını karşılayan bir web sitesidir. Proje için veri seti araştırması sırasında da bizim için uygun veri setini bulmamıza yardımcı olmuştur. Ayrıca veri setleri üzerinde yapılan çalışmalar mevcutsa bu çalışmaları da geliştiricilere paylaşıyor ve aynı zamanda geliştiricilerin de kendi modellerini paylaşmasını mümkün kılıyor. Bizler projemizin gerçekleştirileceği veri seti için yapılmış çalışmaları araştırdık ve de bu konuda bizim kullanacağımız algoritmaları kullanarak bir çalışma yapmış olan bir geliştiriciyi referans aldık [1].

Çalışmada 4 adet makine öğrenmesi algoritması kullanılmıştır. Bunlar Logistic Regression, K-Nearest Neighbors, Decision Tree ve Random Forest allgoritmalarıdır. Logistic regression algoritması sonucu elde edilen doğruluk oranı 0.954 olarak hesaplanmıştır.

Bir sonraki adımda K-Nearest Neighbors Algoritması ile makine öğrenmesi modeli tasarlanmış ve sonuçlar incelenmiştir. Ardından komşu sayıları, algoritmayı optimize edecek şekilde yeniden düzenlenmiştir ve bu karşılaştırma grafik halinde bizlere sunulmuştur. Şekil 2.3'te de görüldüğü gibi 1'den 9'a kadar olan komşu sayıları sürekli güncellenip model eğitildiğinde eğitilen verinin doğruluk oranı 0.94'e, test edilen verinin doğruluk oranı 0.93 değerine yakınsamaktadır ve sabitlenmektedir. Bir sonraki algoritma olan Decision-Tree Algoritması kullanılarak bir model tasarlanmıştır. Bu model incelendiğinde 0.968 oranında doğruluk oranı elde edilmiştir. Son olarak ise veriler Random Forest Algoritması ile tasarlanmış bir model aracılığıyla tahmin

```
In [25]: # Logistic regression
lr = LogisticRegression(random_state=0)
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)

# Get the accuracy score
acc=accuracy_score(y_test, y_pred)

# Append to the accuracy list
accuracy_lst.append(acc)

print("[Logistic regression algorithm] accuracy_score: {:.3f}.".format(acc))
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:469: FutureWarning:
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to sil
ence this warning.
    "this warning.", FutureWarning)

[Logistic regression algorithm] accuracy_score: 0.954.
```

Şekil 2.1 Logistic Regression Doğruluk Oranı

ettirilmiştir ve de 0.974 doğruluk oranına ulaşılmıştır. Makine öğrenmesi eğitimleri sonrasında Severity değerini en çok etkileyen parametreler Şekil 2.6'de gösterilmiştir. Elde edilen top-k listesine göre random forest algoritması ise Şekil 2.9'da da gösterildiği üzere 0.973 doğruluk oranına ulaşmıştır. Son olarak 4 adet algoritmanın doğruluk oranlarının karşılaştırılmış ve Şekil 2.10'deki grafikte gösterilmiştir. Uyarı olarak şunu belirtmeliyiz ki bu yapılan çalışma 1 yıl öncesine ait. Bu veri setine 1 yıl önceinde 2 milyon 500 bine yakın kayıt işlenmişken, mevcut halinde 3 milyon 500 bine yakın kayıt işlenmiş olduğu unutulmamalıdır.

```
In [26]: # Create a k-NN classifier with 6 neighbors
knn = KNeighborsClassifier(n_neighbors=6)

# Fit the classifier to the data
knn.fit(X_train,y_train)

# Predict the labels for the training data X
y_pred = knn.predict(X_test)

# Get the accuracy score
acc=accuracy_score(y_test, y_pred)

# Append to the accuracy list
accuracy_lst.append(acc)

print('[K-Nearest Neighbors (KNN)] knn.score: {:.3f}'.format(knn.score(X_test, y_test)))
print('[K-Nearest Neighbors (KNN)] accuracy_score: {:.3f}'.format(acc))
```

[**K-Nearest Neighbors (KNN)**] knn.score: 0.935.
[**K-Nearest Neighbors (KNN)**] accuracy_score: 0.935.

Şekil 2.2 K-Nearest Neighbors Algoritması Doğruluk Oranı

```
In [27]: # Setup arrays to store train and test accuracies
neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over different values of k
for i, n_neighor in enumerate(neighbors):

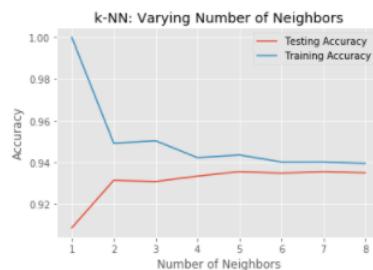
    # Setup a k-NN Classifier with n_neighor
    knn = KNeighborsClassifier(n_neighbors=n_neighor)

    # Fit the classifier to the training data
    knn.fit(X_train,y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)

    #Compute accuracy on the testing set
    test_accuracy[i] = knn.score(X_test, y_test)

# Generate plot
plt.title('k-NN: Varying Number of Neighbors')
plt.plot(neighbors, test_accuracy, label = 'Testing Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training Accuracy')
plt.legend()
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.show()
```



Şekil 2.3 Optimize Edilmiş K-Nearest Neighbors Algoritması

```
In [28]:
# Decision tree algorithm

# Instantiate dt_entropy, set 'entropy' as the information criterion
dt_entropy = DecisionTreeClassifier(max_depth=8, criterion='entropy', random_state=1)

# Fit dt_entropy to the training set
dt_entropy.fit(X_train, y_train)

# Use dt_entropy to predict test set labels
y_pred= dt_entropy.predict(X_test)

# Evaluate accuracy_entropy
accuracy_entropy = accuracy_score(y_test, y_pred)

# Print accuracy_entropy
print('[Decision Tree -- entropy] accuracy_score: {:.3f}'.format(accuracy_entropy))

# Instantiate dt_gini, set 'gini' as the information criterion
dt_gini = DecisionTreeClassifier(max_depth=8, criterion='gini', random_state=1)

# Fit dt_entropy to the training set
dt_gini.fit(X_train, y_train)

# Use dt_entropy to predict test set labels
y_pred= dt_gini.predict(X_test)

# Evaluate accuracy_entropy
accuracy_gini = accuracy_score(y_test, y_pred)

# Append to the accuracy list
acc=accuracy_gini
accuracy_lst.append(acc)

# Print accuracy_gini
print('[Decision Tree -- gini] accuracy_score: {:.3f}'.format(accuracy_gini))

[Decision Tree -- entropy] accuracy_score: 0.967.
[Decision Tree -- gini] accuracy_score: 0.968.
```

Şekil 2.4 Decision Tree Algoritması Doğruluk Oranı

```
n_estimators=100

29]:
# Random Forest algorithm

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)

# Get the accuracy score
acc=accuracy_score(y_test, y_pred)

# Append to the accuracy list
accuracy_lst.append(acc)

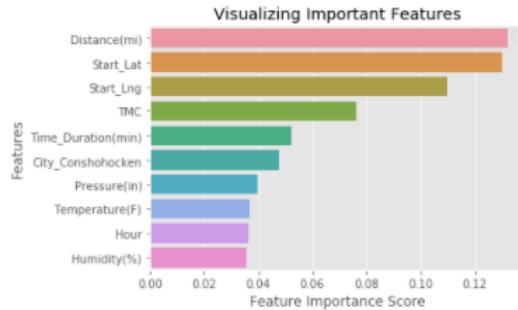
# Model Accuracy, how often is the classifier correct?
print("[Random forest algorithm] accuracy_score: {:.3f}.".format(acc))

[Random forest algorithm] accuracy_score: 0.974.
```

Şekil 2.5 Random Forest Algoritması Doğruluk Oranı

```
In [30]: feature_imp = pd.Series(clf.feature_importances_, index=X.columns).sort_values(ascending=False)

# Creating a bar plot, displaying only the top k features
k=10
sns.barplot(x=feature_imp[:10], y=feature_imp.index[:k])
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```



Şekil 2.6 Severity Değerini En Çok Etkileyen 10 Faktör

```
In [31]: # List top k important features
k=20
feature_imp.sort_values(ascending=False)[:k]
```

```
Out[31]:
Distance(mi)      0.132071
Start_Lat         0.130219
Start_Lng         0.109706
TMC               0.076326
Time_Duration(min) 0.052122
City_Conshohocken 0.047735
Pressure(in)       0.039958
Temperature(F)     0.037064
Hour               0.036530
Humidity(%)        0.035586
Traffic_Signal     0.023557
Side_R             0.020803
City_Bala_Cynwyd  0.014978
Junction           0.014384
City_King_of_Pruessia 0.014276
Visibility(mi)      0.013930
City_Plymouth_Meeting 0.011725
City_Narberth       0.009232
Weekday_Sun          0.007879
Weather_Condition_Overcast 0.007580
dtype: float64
```

Şekil 2.7 Severity Değerini En Çok Etkileyen k Faktör Sıralaması

```

# Create a selector object that will use the random forest classifier to identify
# features that have an importance of more than 0.03
sfm = SelectFromModel(clf, threshold=0.03)

# Train the selector
sfm.fit(X_train, y_train)

feat_labels=X.columns

# Print the names of the most important features
for feature_list_index in sfm.get_support(indices=True):
    print(feat_labels[feature_list_index])

```

TMC
Start_Lng
Start_Lat
Distance(mi)
Temperature(F)
Humidity(%)
Pressure(in)
Hour
Time_Duration(min)
City_Conshohocken

In [33]:

```

# Transform the data to create a new dataset containing only the most important features
# Note: We have to apply the transform to both the training X and test X data.
X_important_train = sfm.transform(X_train)
X_important_test = sfm.transform(X_test)

# Create a new random forest classifier for the most important features
clf_important = RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1)

# Train the new classifier on the new dataset containing the most important features
clf_important.fit(X_important_train, y_train)

```

Out[33]:

```

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=100,
                      n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                      warm_start=False)

```

Şekil 2.8 Severity Değerini En Çok Etkileyen k Faktöre Göre Verinin Düzenlenmesi ve Train-Test Ayrımı

```
In [34]: # Apply The Full Featured Classifier To The Test Data
y_pred = clf.predict(X_test)

# View The Accuracy Of Our Full Feature Model
print('[Random forest algorithm -- Full feature] accuracy_score: {:.3f}'.format(accuracy_score(y_test, y_pred)))

# Apply The Full Featured Classifier To The Test Data
y_important_pred = clf_important.predict(X_important_test)

# View The Accuracy Of Our Limited Feature Model
print('[Random forest algorithm -- Limited feature] accuracy_score: {:.3f}'.format(accuracy_score(y_test, y_important_pred)))

[Random forest algorithm -- Full feature] accuracy_score: 0.974.
[Random forest algorithm -- Limited feature] accuracy_score: 0.973.
```

Şekil 2.9 Severity Değerini En Çok Etkileyen k Faktöre Göre Modelin Random Forest Algoritması İle Eğitilmesi

```
In [35]: # Make a plot of the accuracy scores for different algorithms

# Generate a list of ticks for y-axis
y_ticks=np.arange(len(algo_lst))

# Combine the list of algorithms and list of accuracy scores into a dataframe, sort the value based on accuracy score
df_acc=pd.DataFrame(list(zip(algo_lst, accuracy_lst)), columns=['Algorithm','Accuracy_Score']).sort_values(by=['Accuracy_Score'], ascending = True)

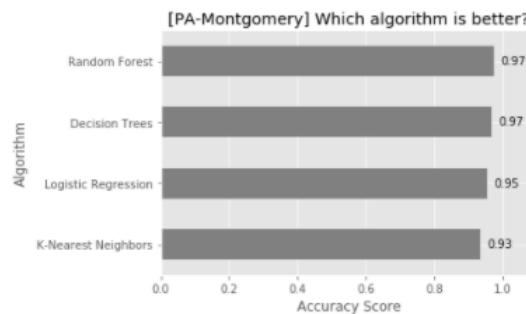
# Export to a file
df_acc.to_csv('./Accuracy_scores_algorithms_{}.csv'.format(state), index=False)

# Make a plot
ax=df_acc.plot.barh('Algorithm', 'Accuracy_Score', align='center', legend=False, color='0.5')

# Add the data label on to the plot
for i in ax.patches:
    # get_width pulls left or right; get_y pushes up or down
    ax.text(i.get_width()+0.02, i.get_y()+0.2, str(round(i.get_width(),2)), fontsize=10)

# Set the limit, labels, ticks and title
plt.xlim(0,1.1)
plt.xlabel('Accuracy Score')
plt.yticks(y_ticks, df_acc['Algorithm'], rotation=0)
plt.title('{0}-{1} Which algorithm is better?'.format(state, county))

plt.show()
```



Şekil 2.10 4 Algoritmanın Karşılaştırılması

3

FİZİBİLİTE

Bu bölümde proje gerçekleştirilirken gerçekleştirilen fizibilite adımları sırasıyla anlatılacaktır. Ayrıca fizibilite çalışması anlatılırken aşağıdaki görsellerden yararlanılacaktır.

3.1 Genel Gereksinim Karşılaştırma Tabloları

	PERFORMANS	FİYAT
I7-8750H	YÜKSEK	200\$
RTX 2060	FAZLA YÜKSEK	450\$

Tablo 3.1 İşlemci

	HİZ	FİYAT
WINDOWS	HIZLI	200\$
LINUX	ÇOK HIZLI	ÜCRETSİZ

Tablo 3.2 İşletim Sistemi

	PERFORMANS	FİYAT
2GB	YÜKSEK	50\$
4GB	YÜKSEK	150\$
16GB	ÇOK YÜKSEK	450\$

Tablo 3.3 RAM

	BAŞARI	FİYAT
PYTHON	YÜKSEK	ÜCRETSİZ

Tablo 3.4 Yazılım Dili

	HIZ	BAŞARI
DECISION TREE ALGORITHM	YÜKSEK	YÜKSEK
RANDOM FOREST ALGORITHM	YÜKSEK	YÜKSEK
NAIVE BAYES ALGORITHM	YÜKSEK	YÜKSEK

Tablo 3.5 Algoritma

	KULLANILIRLIK	FİYAT
ANACONDA	ÇOK YÜKSEK	ÜCRETSİZ
PYCHARM	YÜKSEK	ÜCRETSİZ

Tablo 3.6 Geliştirme Ortamı

3.2 Teknik Fizibilite

Teknik fizibilite, marka bağımsız teknik özelliklerin vurgulandığı bir çalışma olup kendi içinde Yazılım Fizibilitesi ve Donanım Fizibilitesi şeklinde gruplanmıştır.

3.2.1 Yazılım Fizibilitesi

Tablo 3.2 , 3.4 , 3.5 ve 3.6 'den yararlanılarak bir takım sistem ihtiyaçlarına şu şekilde karar verilmiştir. Doğruluk oranı ve hız bizim için önemli iki veri olduğundan işletim sistemi olarak Windows tercih edilmiştir. Yazılım geliştirirken bize sunacağı rahatlık ve kolaylıklardan dolayı anaconda geliştirme ortamı proje boyunca kullanılacaktır. Yazılım dili olarak efektiflik ve hız veriminden dolayı python tercih edilmiş, python'ın başarısı bizim araştırmamız için yeterli olacaktır. Tablo 3.5'den de görüleceği üzere algoritmaların tümü incelenmiş ve araştırma projemize en uygun algoritma olarak belirlenmiştir.

3.2.2 Donanım Fizibilitesi

Tablo 3.1 , 3.2 ve 3.3 'den yararlanılarak donanım unsurlarına şu şekilde karar verilmiştir. Proje bütçesinde, bizi sıkıntıya sokmayacak ve bir o kadar da

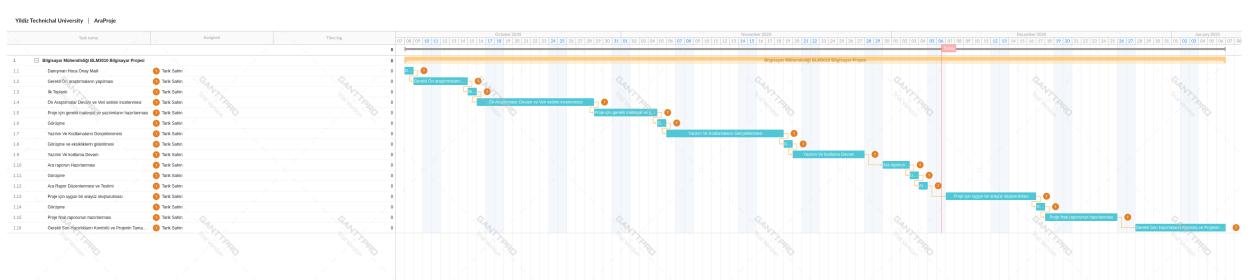
İhtiyaçlarımızı karşılayacak bir sistem kurulumu için gerekli harcamalar yapılmıştır. 16GB RAM, Windows İşletim Sistemi, I7-8750H işlemci bu projede kullanılacak donanım elemanlarıdır.

3.3 İş gücü ve Zaman Planlaması

3.3.1 Zaman Fizibilitesi Ve Gantt Diyagramı

Projeye ait olan Gantt diyagramı aşağıda belirtildiği gibidir.

Görevi gerçekleştiren isim "Tarik Sahin" olarak görünse de bir grup olunduğundan "Tarik Sahin" proje grubu olarak düşünülmelidir.



Şekil 3.1 Gantt Diyagramı

Proje danışman hocamız olan Sayın Furkan Çakmak ile her iki haftada bir Perşembe günleri toplanılmış ve proje için yapılabilecek ve geliştirilebilecek adımlar görüşülmüştür.

3.4 Yasal Fizibilite

Projemiz 5846 Numaralı , Fikir ve Sanat Eserleri Kanunu ile korunan hiç bir sanat veya fikir eserinin haklarını gaspetmemiştir.

3.5 Ekonomik Fizibilite

Ekonominik olarak 16GB RAM (450\$) , I7-8750H işlemci (200\$) , WINDIWS İşletim Sistemi (200\$) toplamda bize 250\$ maliyet çıkarmıştır. Proje genel olarak eğitim amaçlı olarak sunulacağı için bir maddi getirisi olmayacağı için bir amortı süresi de bulunmamaktadır. Sağladığımız en büyük gelir projemiz sayesinde kendisini bir adım ileri götürebilecek insanlardır.

4

Sistem Analizi

Bu bölümde oluşturulan sistemin analizi, sistemde kullanılan modüller açıklanmıştır. Ayrıca algoritmaların işleyişi ve detaylıca çalışma mantığı anlatılmıştır.

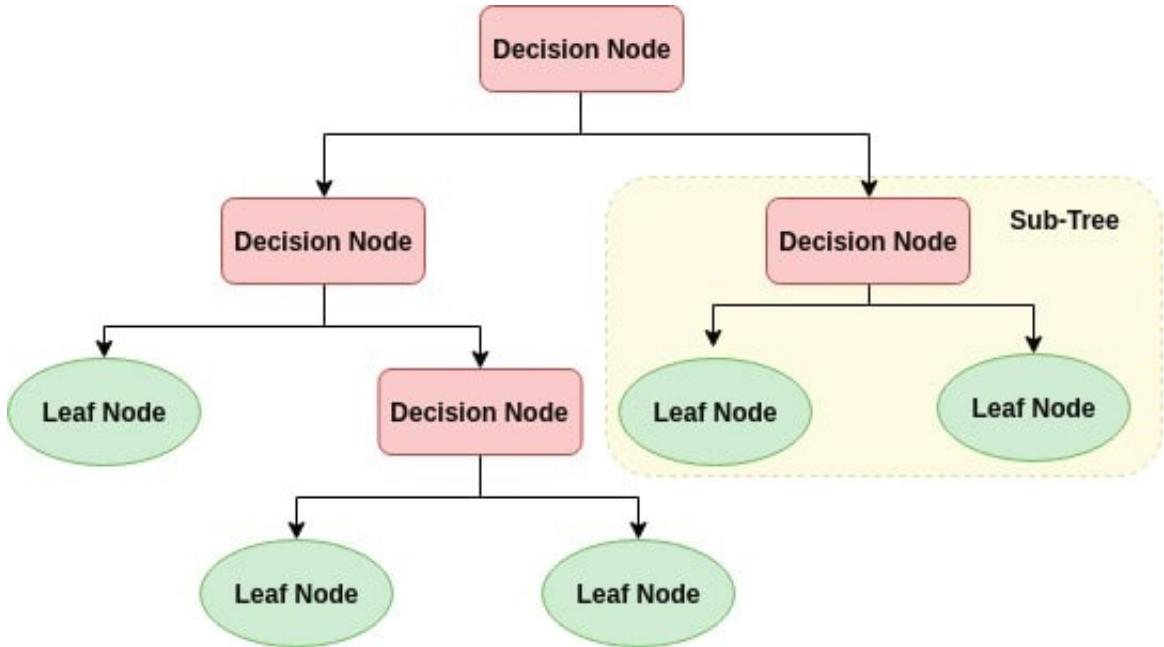
4.1 Kullanılan Araştırma ve Bilgi Toplama Yöntemleri

Trafik kaza şiddeti tahmini için gerekli veri seti kaggle üzerinden edinilmiştir. [2] Python ve projede kullanılan ilgili kütüphanelerin gerekli dökümanları incelenmiş olup, algoritmaların incelenmesi ve araştırmaları aşağıdaki gibidir

4.2 Sistem Modülleri

4.2.1 DTA Model Modülü

[3] Modülü incelemek için öncelikle karar ağacı teriminin ne demek olduğunu iyi kavramamız gereklidir. Karar ağacı, bir iç düğümün özelliği (veya özniteliği) temsil ettiği, dalın bir karar kuralını temsil ettiği ve her bir yaprak düğümün sonucu temsil ettiği akış şemasına benzer bir ağaç yapısıdır. Bir karar ağacındaki en üst düğüm, kök düğüm olarak bilinir. Öznitelik değerine göre bölmeyi öğrenir. Ağacı, özyinelemeli bölmeye çağrı ile özyinelemeli olarak böler. Bu akış şeması benzeri yapı, karar vermede size yardımcı olur. İnsan düzeyindeki düşünceyi kolayca taklit eden bir akış şeması gibi görselleştirmedir. Bu nedenle karar ağaçlarının anlaşılması ve yorumlanması kolaydır.



Şekil 4.1 DTA Sınıflandırma Mantığı

DTA 'nın arkasındaki en temel çalışma mantığı şu şekildedir. Verinin bölünmesi için Attribute Selection Measures (ASM) kullanılarak en iyi attribute değeri seçilir. Seçilen bu attribute değeri bir karar düğümü yapılır ve veri kümesi daha küçük alt kümelere bölünür. Bu bölme işlemi kalan tüm tuple değerleri aynı düğümü göstermeye başlayana kadar veya attribute değerleri bitinceye kadar ya da örnekler bitinceye kadar rekürsif olarak devam eder.

4.2.2 ASM (Attribute Selection Measures Nedir?)

[3] Verileri mümkün olan en iyi şekilde bölümleyen bölme ölçütünü seçmek için kullanılan bulușsal bir yöntemdir. Belirli bir düğümdeki tuplelar için kesme noktalarının belirlenmesine yardımcı olduğu için bölme kuralları olarak da bilinir. Verilen veri setini açıklayarak her özelliğe (veya attribute değerine) bir derece verilir. En iyi dereceye sahip attribute değeri, kaynak olarak seçilir. En popüler seçim ölçütleri Information Gain, Gain Ratio, ve Gini Index'tir.

Information Gain: Entropideki azalmadır. Bölünmeden önceki entropi ile veri kümesinin bölünmesinden sonraki ortalama entropi arasındaki farkı, verilen attribute değerlerine göre hesaplar. ID3 (Iterative Dichotomiser) DTA, Information Gain seçim ölçüsünü kullanır.

Gini Index: Her attribute için bir binary bölünmeyi dikkate alır. Her bölümün safsızlığının ağırlıklı toplamı hesaplanabilir.

Belirtilen formüllerde P_i , D 'deki keyfi bir tuple'ın C_i sınıfına ait olma olasılığıdır.

$$\text{Info}(D) = - \sum_{i=1}^m p_i \log_2 p_i$$

$$\text{Info}_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times \text{Info}(D_j)$$

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

Şekil 4.2 Information Gain Hesabı

Gain Ratio: Split Info'yu kullanarak ve Information Gain'i normalleştirecek sapma sorununu ele alır.

$$\text{SplitInfo}_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right)$$

$$\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}_A(D)}$$

Şekil 4.3 Split Info ve Gain Ratio değerinin hesaplanması

$$Gini(D) = 1 - \sum_{i=1}^m P_i^2$$

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

Şekil 4.4 Gini hesabı

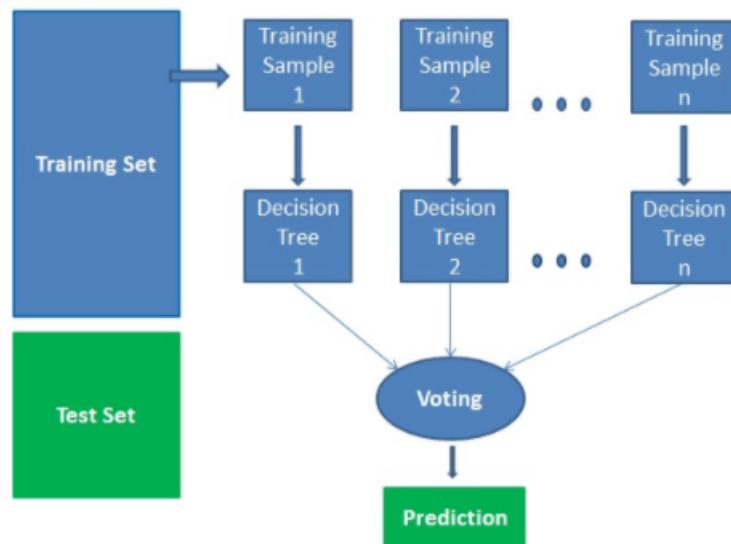
4.2.3 RFA Model Modülü

[4] Diyelim ki arkadaşlarınıza sormaya karar verdiniz ve onlarla çeşitli yerlere geçmiş seyahat deneyimleri hakkında konuştuğunuz. Her arkadaşınızdan bazı tavsiyeler alacaksınız. Şimdi bu önerilen yerlerin bir listesini yapmanız gerekiyor. Ardından, yaptığınız önerilen yerler listesinden oy vermelerini (veya seyahat için en iyi yeri seçmelerini) istersiniz. En fazla oyu alan yer, seyahat için son seçiminiz olacak.

Yukarıdaki karar sürecinde iki bölüm var. İlk olarak, arkadaşlarınıza bireysel seyahat deneyimleri hakkında sorular sorun ve ziyaret ettikleri birden çok yerden tek bir öneri alın. Bu kısım, DTA kullanmak gibidir. Burada her arkadaş şimdiye kadar ziyaret ettiği yerlerin bir seçimini yapar.

Tüm tavsiyeleri topladıktan sonraki ikinci kısım, tavsiyeler listesindeki en iyi yeri seçmek için oylama prosedürüdür. Arkadaşlardan tavsiye alma ve en iyi yeri bulmak için onlara oy verme işleminin tamamı RFA olarak bilinir. RFA, 4 adımda işlem görür. Bu adımlar aşağıda belirtildiği gibidir.

1. Veri setinden rastgele örnekler seçilir.
2. Bu elde edilen örneklerden karar ağaçları oluşturulur ve her bir ağaçtan tahmin sonucu elde edilir.
3. Her bir tahmin sonucu için oylama oluşturulur.
4. En çok oylamaya sahip tahmin sonucu, final tahmin değeri olarak hesaplanır.



Şekil 4.5 RFA Çalışma Mantığı

Tablo 4.1 RFA Avantaj ve Dezavantajları

AVANTAJLARI	DEZAVANTAJLARI
Yüksek Doğruluk Oranı	Yavaş Çalışma
Overfitting problemi yok	Modeli Yorumlayabilmek çok zor
Regression problemlerinde kullanılabilir	-
Hatalı(Missing) değerler kolayca aşılabilir	-
Sınıflandırma problemlerinde kullanılabilir	-

Tablo 4.1 incelendiğinde RFA ile yüksek doğruluk oranına sahip verilere daha uzun sürede ulaşılabilmekeyken, DFA ile RFA'ya göre daha düşük doğruluk oranına sahip tahminlere daha kısa sürede ulaşılabilmektedir.

4.2.4 NBA Model Modülü

[5] Naive Bayes, Bayes teoremine dayalı bir istatistiksel sınıflandırma tekniğidir. Özellikle büyük boyuta sahip veri setlerinde hızlı, güvenilir ve yüksek doğruluk oranına sahip bir algoritmadır. Naive Bayes, spesifik bir özelliği diğer özelliklerden bağımsız olarak ele alarak işler. Örneğin, bir kredi başvurusu, gelirine, önceki kredisine ve işlem geçmişine, yaşına ve konumuna bağlı olarak istenir veya istenmez. Bu özellikler birbirine bağlı olsa bile, bu özellikler yine de bağımsız olarak değerlendirilir. Bu varsayımdan hesaplamayı basitleştirir ve bu yüzden saf olarak kabul edilir. Bu varsayıma sınıf koşullu bağımsızlık(Class Conditional Independence) denir ve hesaplanma formülü Şekil 4.6'de gösterilmiştir.

Şekil 4.6'daki değişkenler:

- >P(h): h hipotezinin ya da durumunun doğru olma olasılığı
- >P(D): Verinin olasılığı(h hipotezinden ya da durumundan bağımsız olarak)
- >P(h|D): h hipotezi ya da durumunun verilen D datasındaki olasılığı
- >P(D|h): D datasında h hipotezi ya da durumunun doğru olma olasılığı

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Şekil 4.6 Class Conditional Independence Olasılık hesabı

- 1-) Verilen sınıfların etiketlerinin olasıklarını hesaplayın (Örn: Güneşli olma olasılığı)
- 2-) Her bir sınıfın attribute değerinin Likelihood olasılığını hesaplayın.
- 3-) Bayes formülü ile bu elde edilen değerlerin son olasılıklarını hesaplanır.
- 4-) Hangi sınıfın daha yüksek olasılığa sahip olduğu hesaplanır.

The diagram illustrates the process of creating Naive Bayes models. It starts with a raw dataset table on the left, which is then processed into a Frequency Table (top right). This Frequency Table is further processed into two Likelihood Tables: Likelihood Table 1 (middle right) and Likelihood Table 2 (bottom right).

Whether	Play
Sunny	No
Sunny	No
Overcast	Yes
Rainy	Yes
Rainy	Yes
Rainy	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Whether	No	Yes
Overcast		4
Sunny	2	3
Rainy	3	2
Total	5	9

Likelihood Table 1		
Whether	No	Yes
Overcast	4	=4/14 0.29
Sunny	2	=5/14 0.36
Rainy	3	=5/14 0.36
Total	5	9
	=5/14 =9/14	
	0.36	0.64

Likelihood Table 2				
Whether	No	Yes	Posterior Probability for No	Posterior Probability for Yes
Overcast	4	3	0/5=0	4/9=0.44
Sunny	2	3	2/5=0.4	3/9=0.33
Rainy	3	2	3/5=0.6	2/9=0.22
Total	5	9		

Şekil 4.7 Likelihood tabloları oluşturma ve Naive Bayes olasılığının hesaplanması

Tablo 4.2 NBA Avantaj ve Dezavantajları

AVANTAJLARI	DEZAVANTAJLARI
Yüksek Doğruluk Oranı	Bağımsız özelliklerin tahmini daha doğru
Hızlı çalışma	Zero Probability/Frequency Problemi
Düşük hesaplama maliyeti	Hatalı(Missing) değerler aşılamaz
Metin analizinde yüksek verim	-
Çoklu sınıf problemlerinde verimli	-
Daha az yorucu	-

Tablo 4.2 incelendiğinde NBA daha çok bağımsızlık ilkesiyle oluşturulmuş veri setlerinde en yüksek doğruluk tahmini sonuçlarına ulaşırken genel olarak hızlı ve daha az yorucu çalışan bir algoritmadır.

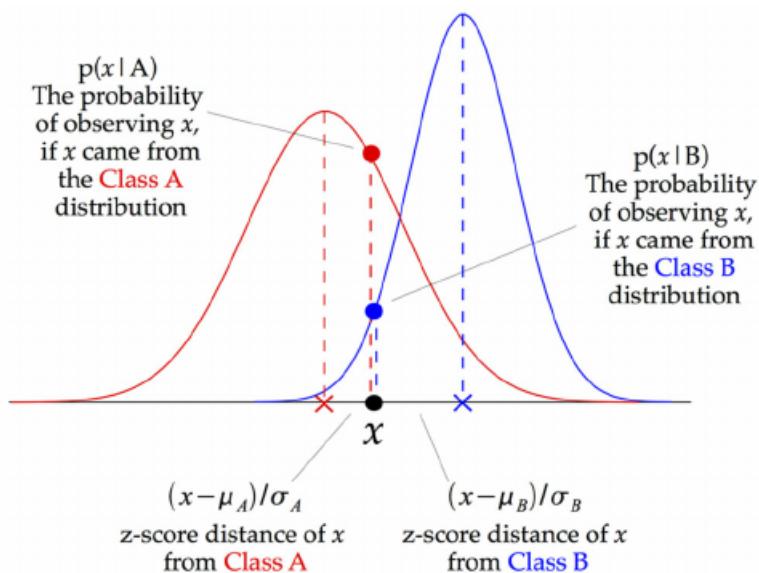
4.2.5 Gaussian NBA Model Modülü

[6] Gaussian NBA, Gauss normal dağılımını takip eden ve sürekli verileri destekleyen bir Naive Bayes Algoritması çeşididir. Sürekli verilerle çalışırken, sıkılıkla alınan bir varsayımdır, her sınıfla ilişkili sürekli değerlerin normal (veya Gauss) bir dağılıma göre dağıtımının gerçekleştirildiğidir. Naive Bayes'in özünde Şekil 4.6'da gösterildiği formül ile Likelihood tabloları oluştururken Gaussian Navie Bayes'in özünde Likelihood tablosundaki verilerin özelliklerin olasılığı Şekil 4.8'deki formüle göre hesaplanır.

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Şekil 4.8 Gaussian NBA Olasılık Hesabı

Şekil 4.9'da ise Gaussian NBA'daki verilerin sınıflandırılmasındaki dağılım gösterilmiştir. Her veri noktasında, o nokta ile her sınıf ortalaması arasındaki z-score distance hesaplanır, yani veri noktasının sınıf ortalamasından uzaklığının o sınıfın standart sapmasına bölünmesidir.



Şekil 4.9 Gaussian NBA Sınıflandırma

4.2.6 Bernoulli NBA Model Modülü

[7] Bernoulli NBA, Bernoulli dağılımını takip eden ve ayrık verileri destekleyen bir Naive Bayes Algoritması çeşididir. Bernoulli Naive Bayes'in temelinde, özellikleri yalnızca doğru veya yanlış, evet veya hayır, başarı veya başarısızlık, 0 veya 1 gibi ikili değerler olarak kabul etmesi yatar. Bu nedenle, özellik değerleri ikili olduğunda, Bernoulli Naive Bayes sınıflandırıcısını kullanmamız gerektiğini biliyoruz. Bernoulli temel olasılık dağılımı Şekil 4.10 'da gösterildiği gibidir.

The Bernoulli distribution

$$p(x) = P[X = x] = \begin{cases} q = 1 - p & x = 0 \\ p & x = 1 \end{cases}$$

Şekil 4.10 Bernoulli NBA Olasılık Hesabı

Şekil 4.11'de ise Bernoulli NBA'daki verilerin sınıflandırılmasındaki temel kural gösterilmektedir.

Bernoulli Naive Bayes Classifier is based on the following rule:

$$P(x_i | y) = P(i | y)x_i + (1 - P(i | y))(1 - x_i)$$

Şekil 4.11 Bernoulli NBA Sınıflandırma

Tablo 4.3 Bernoulli NBA Avantaj ve Dezavantajları

AVANTAJLARI	DEZAVANTAJLARI
Aşırı hızlı çalışma	Hatalı(Missing) değerler aşılamaz.
Özellikler binary değerlerle ele alınır	Veri hassaslığı
Özellikler bağımsız ele alınır	-
Metin sınıflandırmada verimli	Train edilmeyen veri tahmin edilemez
Anlık tahmin problemlerinde verimli	-
İlgisiz(Irrelevant) özellikler kolay aşılır	-
Sonuçlar kolay anlaşılır	-

Tablo 4.3'ten de anlaşılacağı gibi Bernoulli NBA, genel olarak ikili tahmin çeşitlerinin(Evet-Hayır, 1-0, Var-Yok, Kadın-Erkek) gerçekleştirilemesinde ve anlık olarak sonuç üretilmesi istenildiğinde kullanılan bir algoritmadır.

4.2.7 Multinomial NBA Model Modülü

[8] Multinomial NBA, bir terimin skiliğini(frekansını), yani bir terimin veri setinde ne kadar geçtiğini kullanarak tahmin yürütür. Terim sıklığı, genellikle işlenmemiş terim sıklığının belge uzunluğuna bölünmesiyle normalleştirilir. Normalleştirmeden sonra, koşullu olasılığı tahmin etmek için eğitim verilerine dayalı olarak maksimum olasılık tahminlerini hesaplamak için terim sıklığı kullanılabilir. Örnek olarak Spam Mail tespitlerinde kullanılabilir. Spam maillerde en çok geçen kelimeler öğrenilir ve bu kelimeler test verilerinde kontrol edilerek bir mailin spam olup olmadığı tahmin edilebilir.

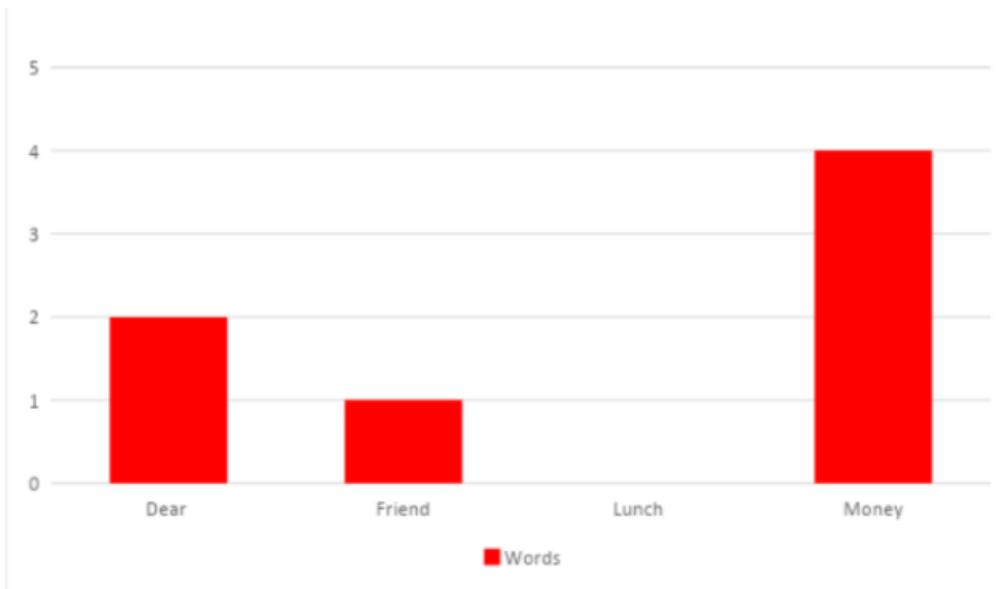
Örnek olarak bir mail adresine gelen **normal** mesajlardaki kelime sıklığı Şekil 4.12'deki histogramdaki gibi olsun.



Şekil 4.12 MultiNomial NBA Normal Mail Örnek

Aynı şekilde bir mail adresine gelen **spam** mesajlardaki kelime sıklığı Şekil 4.13'teki histogramdaki gibi olsun.

Her bir kelimenin geçme sıklığını(frekansını) hesapladığımız zaman elde edilen sonuçlar aslında Likelihood sınıflandırma değerleri olacaktır. "Dera Friend" mesajının spam olup olmadığını hesaplayalım. 8 normal mesaj ve 4 spam mesajımız olsun. $\text{Probability}(\text{Dear}|\text{Normal}) = 8 / 17 = 0.47$ $\text{Probability}(\text{Friend}/\text{Normal}) = 5 / 17$



Şekil 4.13 MultiNomial NBA Spam Mail Örnek

$$=0.29 \text{ Probability (Lunch/Normal)} = 3/ 17 =0.18 \text{ Probability (Money/Normal)} = 1/ 17 =0.06$$

$$\text{Probability (Dear|Spam)} = 2 /7 = 0.29 \text{ Probability (Friend|Spam)} = 1/ 7 =0.14 \\ \text{Probability (Lunch/Spam)} = 0/ 7 =0.00 \text{ Probability (Money/Spam)} = 4/ 7 =0.57$$

"Dear Friend" mesajının normal olma olasılığı = Probability (Dear|Normal)*Probability (Friend|Normal)*Probability (Normal Messages) = $0.67 * 0.47 * 0.29 = 0.09$ olarak hesaplanır.

"Dear Friend" mesajının spam olma olasılığı = Probability (Dear|Spam)*Probability (Friend|Spam)*Probability (Spam Messages) = $0.33 * 0.29 * 0.14 = 0.01$ olarak hesaplanır.

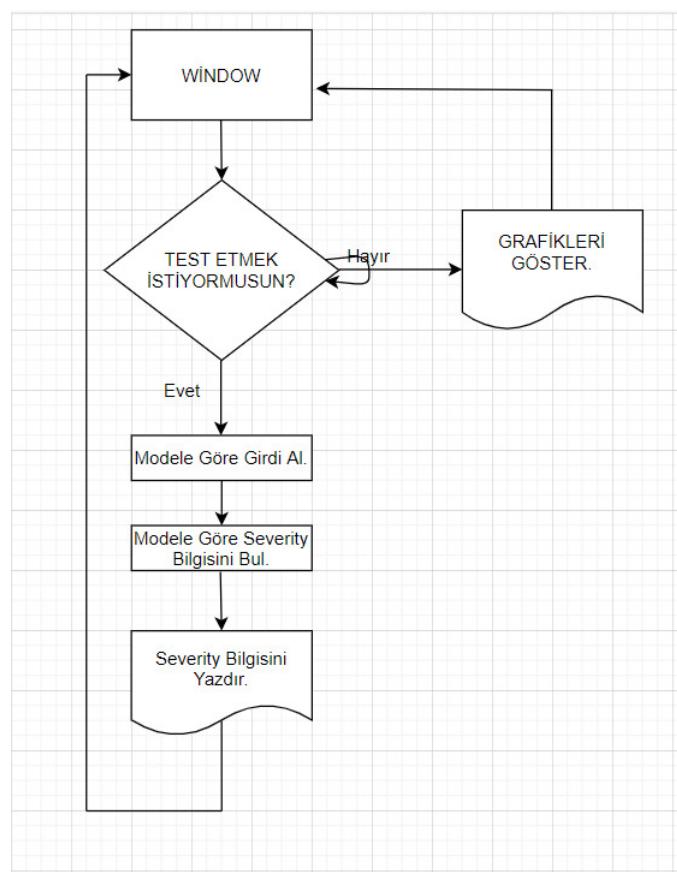
İki sonuç karşılaştırıldığında "Dear Friend " kelime ögesinin spam olmadığı görülür. Temel mantığı bu yapıya göre kurulmuş olan Multinomial NBA, diğer NBA türleri gibi hızlı ve daha yorucu bir şekilde çalışır. Anlık tahminlerde etkili çözümler üretebilir ve de metin sınıflandırmada daha çok kullanılan bir algoritmadır.

5 Sistem Tasarımı

Sistem tasarımı için aşağıda belirtilen diyagramlar başlıklarları ile birlikte verilmiştir.

5.0.1 İş Akış Diyagramı

Şekil 5.1'de iş akış diyagramı gösterilmiştir.



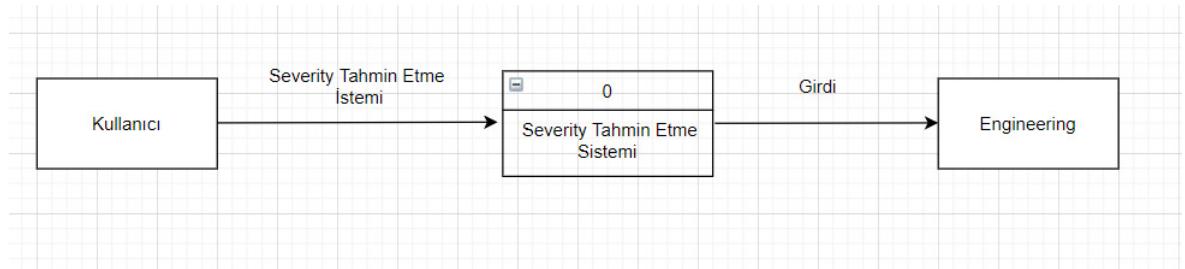
Şekil 5.1 İş akış diyagramı

Kullanıcı sistemi çalıştırıldığından sistem öncelikle 3.5 milyon veriyi belirtilen test boyutuna göre öğrenir. Kullanıcı test etmeden önce ilgili tanımlanmış kaza bilgilerini

grafikler halinde görebilir. Model seçimi sonrasında seçilen modele sistem eğitilir. Sonrasında öğrenilen veri setine göre girilen kaza bilgileri verileri sisteme tahmin ettirilir ve elde edilen sonuçlar kullanıcıya bildirilir.

5.0.2 Context Diyagramı

Şekil 5.2'de Context diyagramı gösterilmiştir.

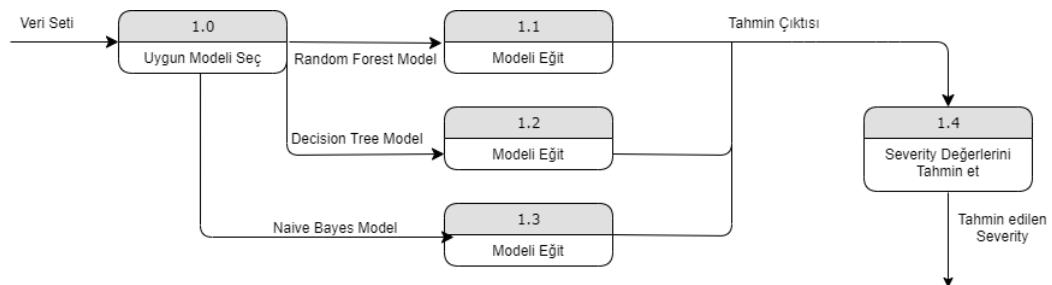


Şekil 5.2 Context Diyagramı

Kullanıcı test komutunu sistemde çalıştırduğunda sistem öncelikle 3.5 milyon veriyi belirtilen test boyutuna ve modele göre öğrenir. Şiddet değerini eğitilen model türüne ve train setine göre tahmin eder ve kullanıcıya bildirir.

5.0.3 Alt Seviye Veri Akış Diyagramı

Şekil 5.3'te Alt Düzey veri akış diyagramı gösterilmiştir.



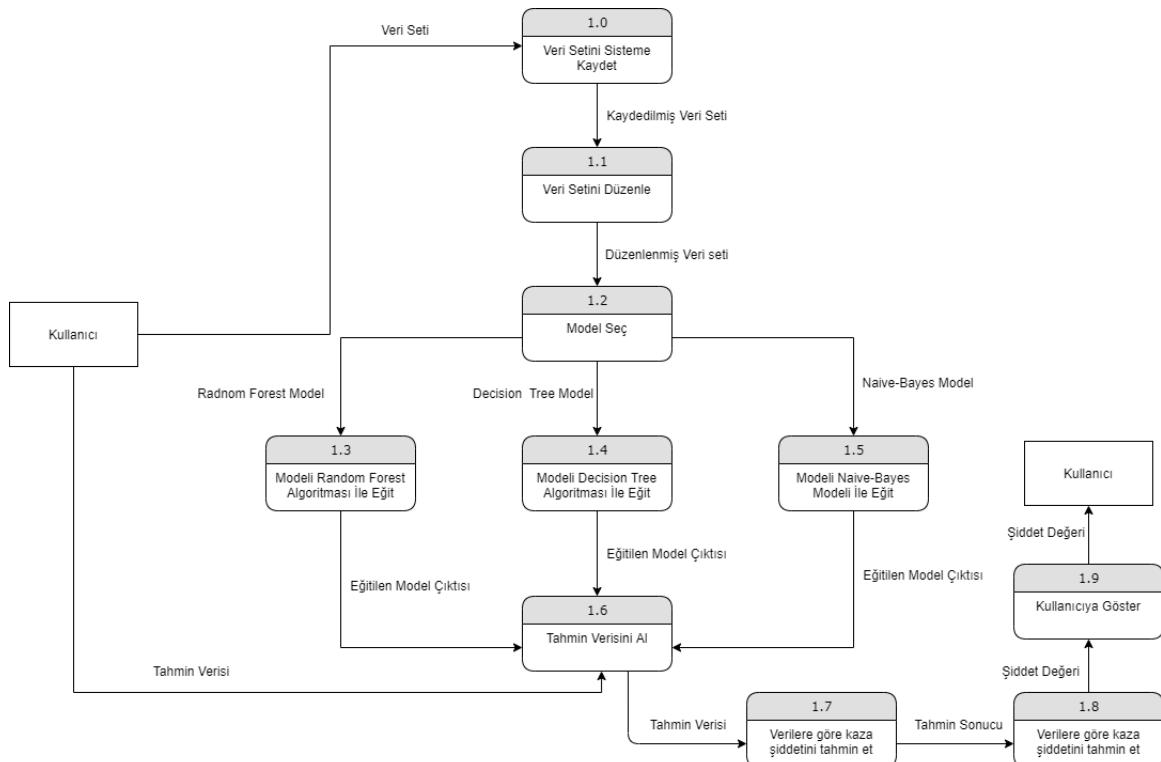
Şekil 5.3 Alt Düzey Veri Akış Diyagramı

Veri setini input alan sistem için öncelikle uygun model seçilir. Ardından seçilen modele göre veri seti öğrenilir. Sonrasında tahmin edilmek istenen veri satırının kaza şiddeti kullanıcıya bildirilir.

5.0.4 0.Düzey Veri Akış Diyagramı

Şekil 5.4'te 0. Düzey veri akış diyagramı gösterilmiştir.

Veri setini kullanıcıdan input alan sistem için öncelikle veri seti uygun şekilde düzenlenir. Sonrasında veri setine en uygun model seçilir. Ardından seçilen modele göre veri seti öğrenilir. Öğrenilen veriler grafikler ve karşılaştırmalar halinde kullanıcıya bildirilir. Sonrasında tahmin edilmek istenen veri satırı kullanıcı tarafından sisteme girilir. Bu girilen veriye göre kaza şiddeti sisteme tahmin ettirilir. Tahmin edilen kaza şiddeti verisi kullanıcıya bildirilir.



Şekil 5.4 0. Düzey Veri Akış Diyagramı

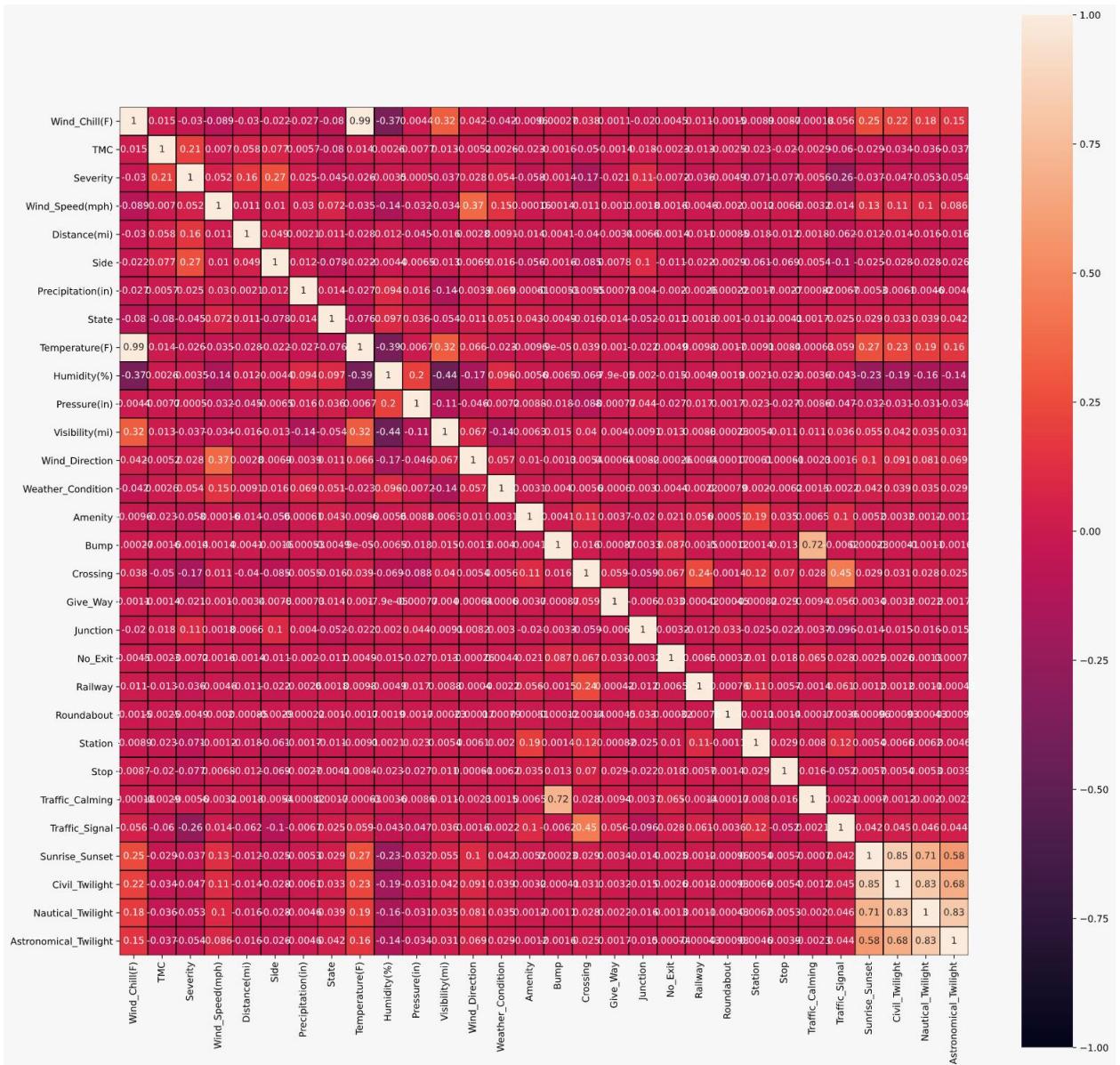
6

Uygulama

Sistem modüllerinde Random Forest Algoritması, Decision Tree Algoritması ve Naive-Bayes Algoritmaları hakkında gerekli olan bilgilere değinilmiştir. Bu bölümde ise her bir algoritma için oluşturulan modellerin tanıtımı ve gerçekleştirilen çalışma anlatılacaktır. Modellerin eğitimine geçilmeden önce veri setilarındaki bilgiler ve veri setinin nasıl düzenlendiği anlatılmıştır.

6.1 Veri Setinin Düzenlenmesi

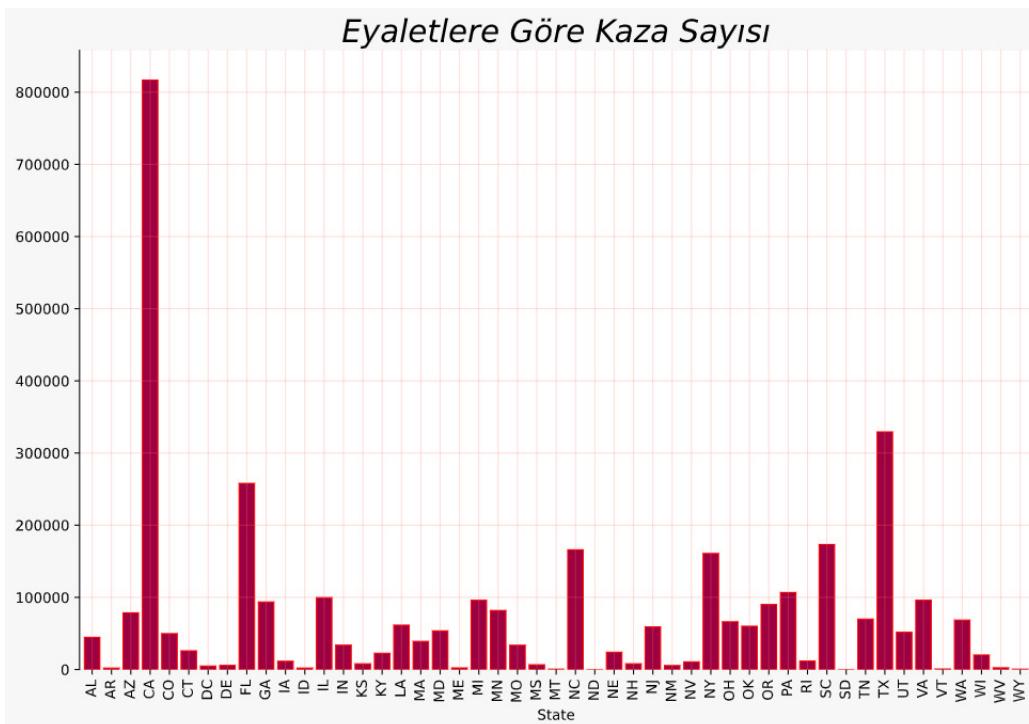
Tüm modeller için ortak olarak kullanılan veri seti "US Accidents Jun20.csv" adıyla kaggle [2] üzerinden temin edilmiştir. İçerisinde yaklaşık olarak 3.5 milyon veri bulunan veri seti, 1.25 GB boyutundadır. Veri setinin her bir satırında 49 adet sütuna yerleştirilmiş veriler bulunmaktadır. Eğitilen modellerin odaklandığı sütunun adı ise "Severity" sütunudur. Her bir satır kaza verilerini içerisinde barındırır denilmiştir, "Severity" kolonunda kazanın şiddeti bulunmaktadır. Bu şiddet değeri (1-4) arasında değerler almaktadır. 1 değeriyle kayda geçmiş kazalar en az şiddete dahil olan kaza, yani trafiğe etkisi en düşük olan kazalar iken 4 değeriyle kayda geçmiş kazalar en şiddetli kaza yani trafiğe etkisi en kötü ve ağır olan kazalardır. Öncelikli olarak verilerin korelasyon haritası Şekil 6.1'de oluşturulmuştur.



Şekil 6.1 Korelasyon Haritası

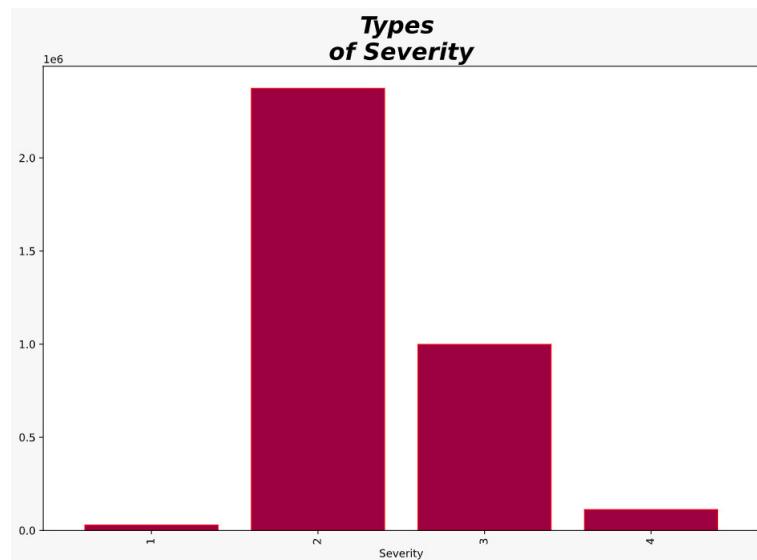
Ayrıca veri setindeki kayıtlı olan eyaletlere göre kazaların sayısı Şekil 6.2'de gösterilmiştir.

Şekil incelediğinde en büyük kaza sayısı CA yani "California" ve TX yani "Texas" eyaletlerinde görülmüştür.

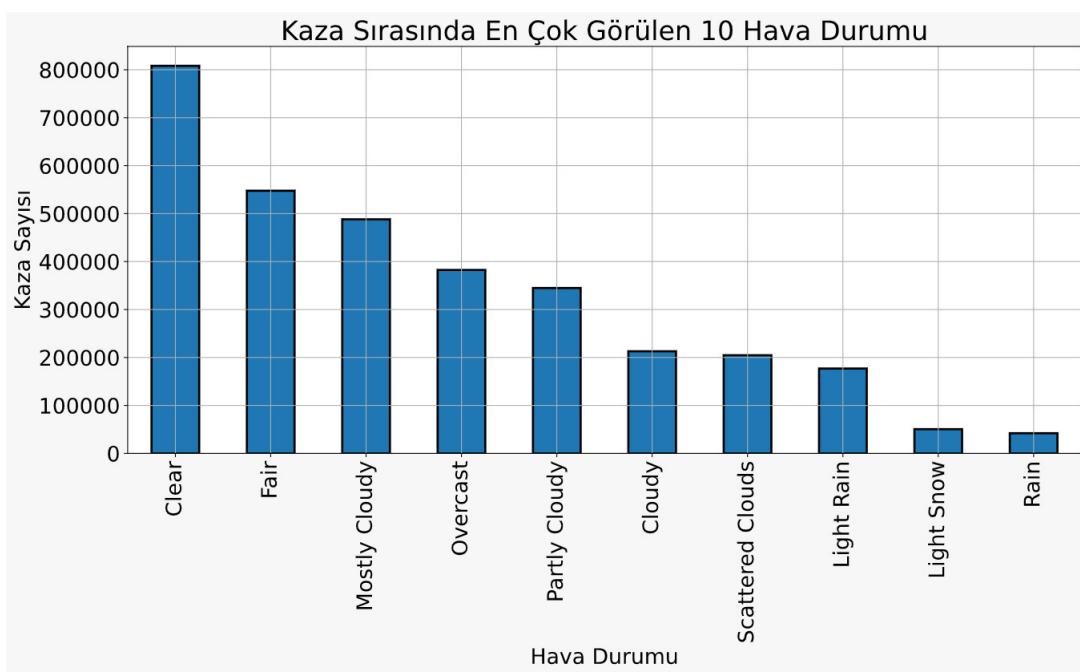


Şekil 6.2 Eyaletlerdeki Kaza Sayıları

Şekil 6.4'te kazaların en çok hangi hava durumu koşulunda gerçekleştiği ve Şekil 6.3'te kaza şiddetinin veri sayısı ayrı ayrı belirtilmiştir. İstistikler incelendiğinde en çok sayıda 2 ve 3 şiddetine ait olan kazalar kayda işlenmişken 1 ve 4 şiddetindeki kazalar nispeten çok daha az kayda geçmiştir. Ayrıca kaza gerçekleştiği sırada ilginç bir şekilde hava durumunda havanın açık olduğu görülmüştür.

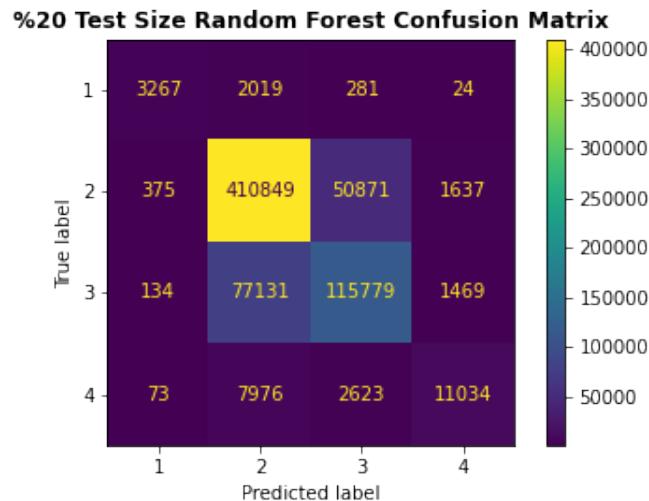


Şekil 6.3 Kaza Şiddeti Çeşitleri ve Sayıları



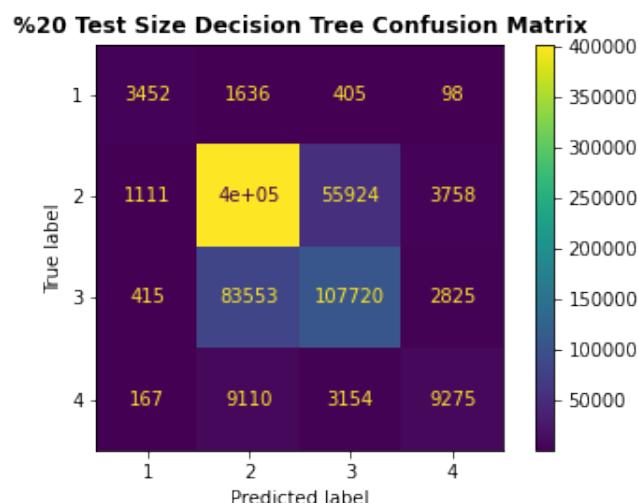
Şekil 6.4 Kazanın En Çok Hangi Hava Koşulunda Gerçekleştiğini Belirten Grafik

Şekil 6.5'deki CM'e göre TP değerler 1 için 3267, yani model ile tüm 1 ler arasından 3267 tanesi doğru tahmin edilmiştir. Aynı şekilde 2 için 410849 adet veri, 3 için 115779 adet veri ve 4 için 11034 adet veri doğru tahmin edilmiştir.



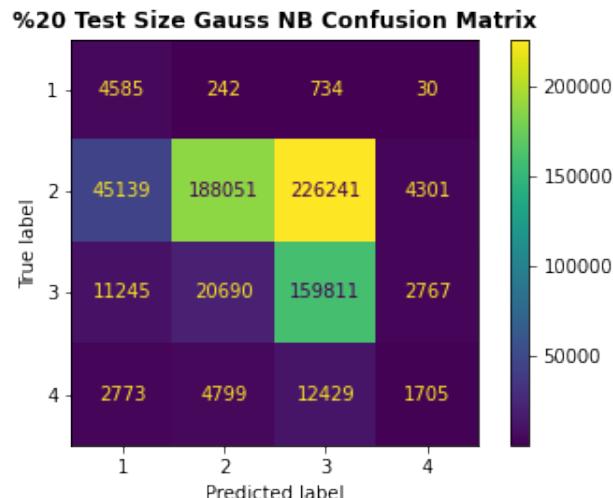
Şekil 6.5 Random Forest % 20 Test Boyutuna Göre CM

Şekil 6.6'deki CM'e göre TP değerler 1 için 3452, yani model ile tüm 1 ler arasından 3452 tanesi doğru tahmin edilmiştir. Aynı şekilde 2 için 4e+05 adet veri, 3 için 107720 adet veri ve 4 için 9275 adet veri doğru tahmin edilmiştir.

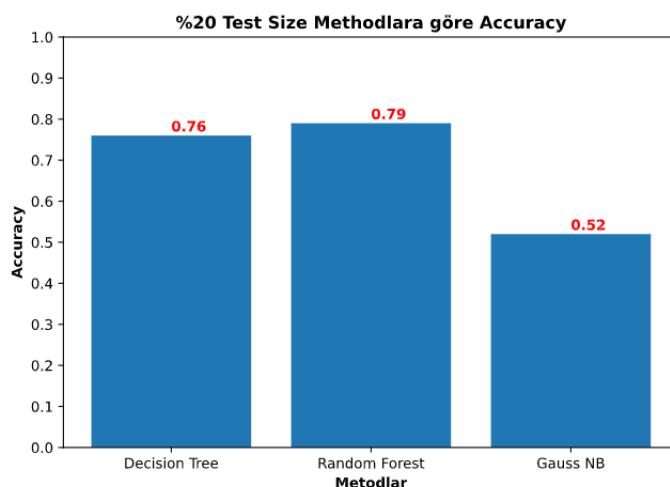


Şekil 6.6 Decision Tree % 20 Test Boyutuna Göre CM

Şekil 6.7'deki CM'e göre TP değerler 1 için 4585, yani model ile tüm 1 ler arasından 4585 tanesi doğru tahmin edilmiştir. Aynı şekilde 2 için 188051 adet veri, 3 için 159811 adet veri ve 4 için 1705 adet veri doğru tahmin edilmiştir.

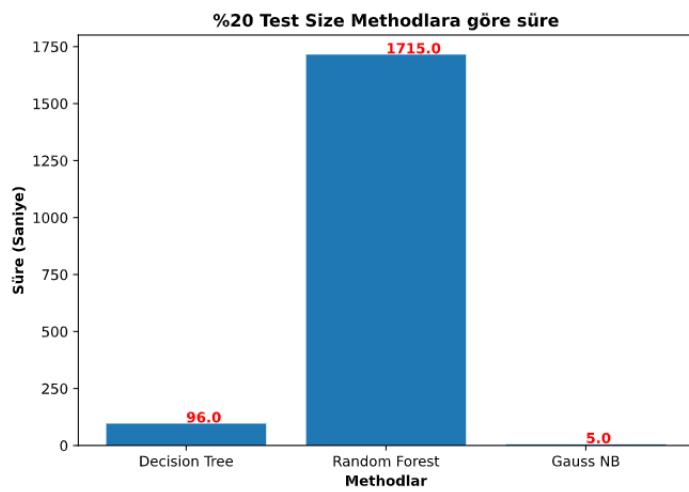


Şekil 6.7 Gaussian Naive Bayes % 20 Test Boyutuna Göre CM

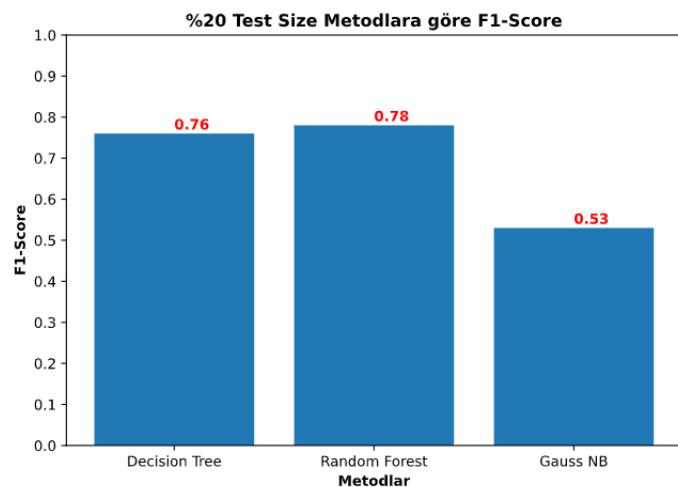


Şekil 6.8 Algoritmaların Doğruluk Değerleri

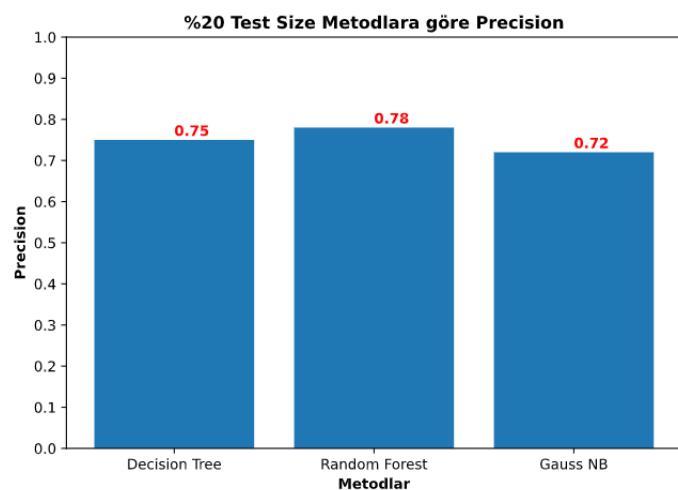
CM'ler ve grafikler incelendiğinde Gaussian Naive Bayes'in en düşük DTA'nın 2. ve RFA'nın en yüksek doğruluk oranına sahip verilere eriştiği görülür. Bu sonuçlar Şekil 6.8'deki grafikten de görülebilir. Fakat hız olarak Şekil 6.9'da incelendiğinde hız olarak en hızlı Gaussian NBA ve en yavaş RFA çalışmaktadır.



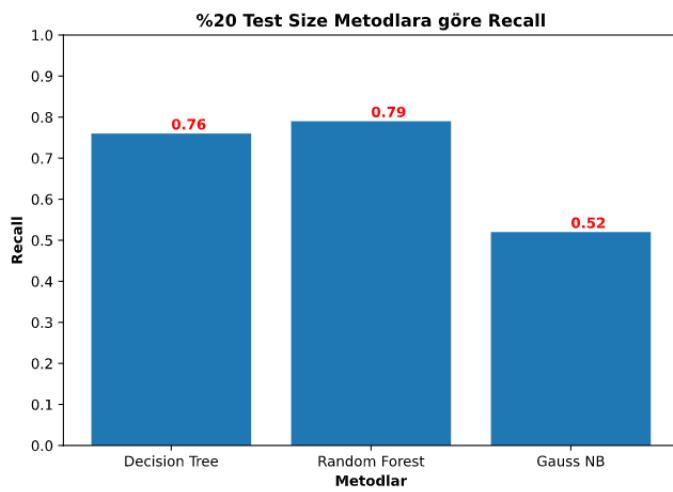
Şekil 6.9 Algoritmaların Çalışma Süreleri



Şekil 6.10 Algoritmaların F-1 Score Değerleri



Şekil 6.11 Algoritmaların Precision Değerleri

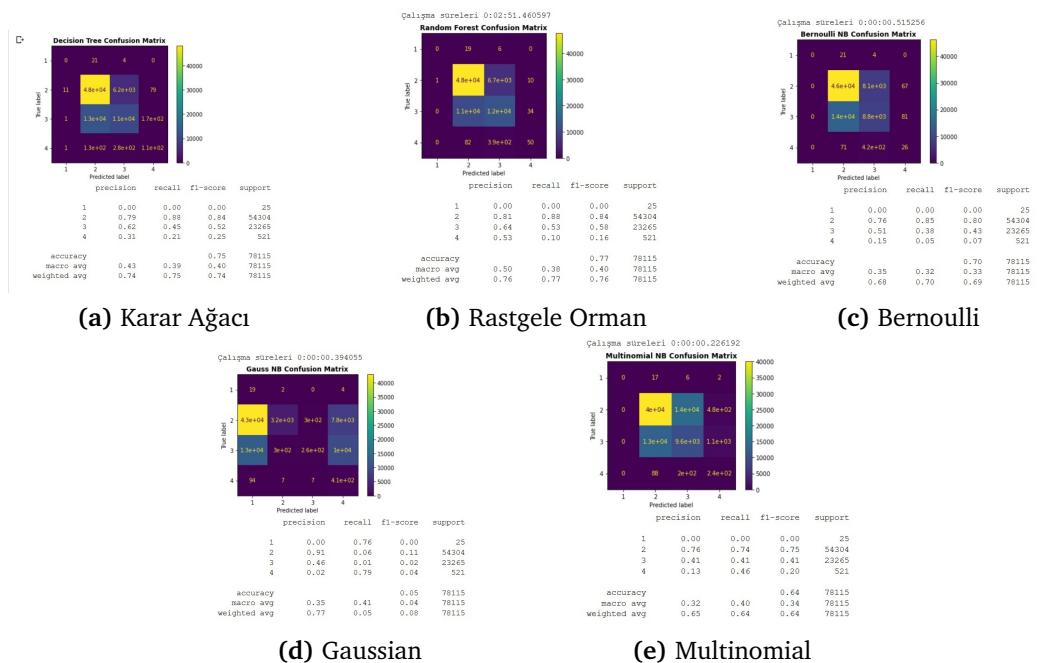


Şekil 6.12 Algoritmaların Recall Değerleri

7

Performans Analizi

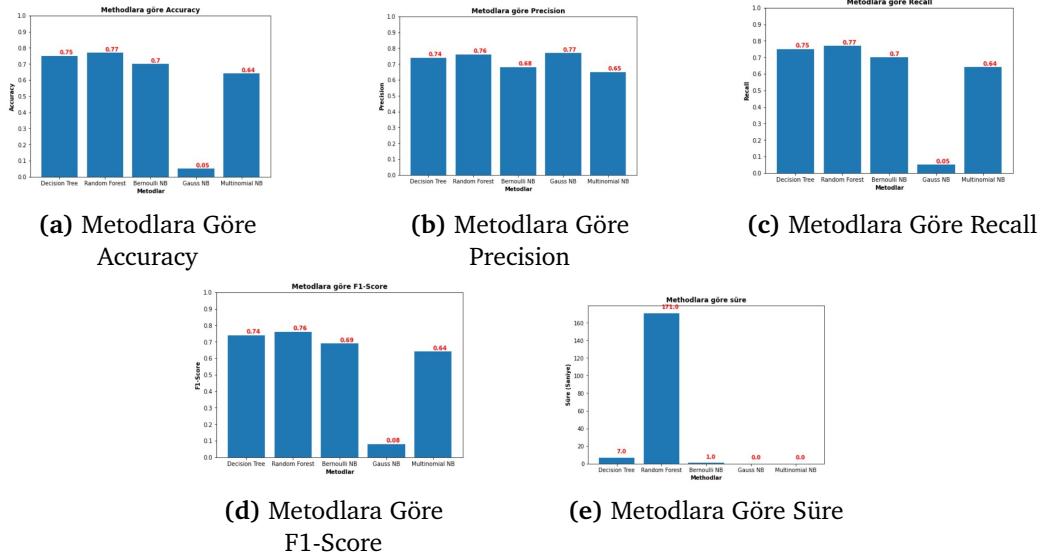
Bu adımda algoritmaların optimize edilmeden önceki durumları incelenmiştir ve gerekli optimizasyon adımları uygulandıktan sonra elde edilen sonuçlar gösterilmiştir. Proje gerçekleştirirken ilk olarak veri seti incelenmiştir. Veri setinde mevcut olan "NaN" değerler ilk akla gelen mantıkla işleme girmemiştir. "NaN" değerlerin veri setinden kaldırılmasıyla aşağıdaki veriler elde edilmiştir.



Şekil 7.1 Test boyutu = 0.1 algoritmaların karmaşıklık matrisleri (Confusion Matrix)

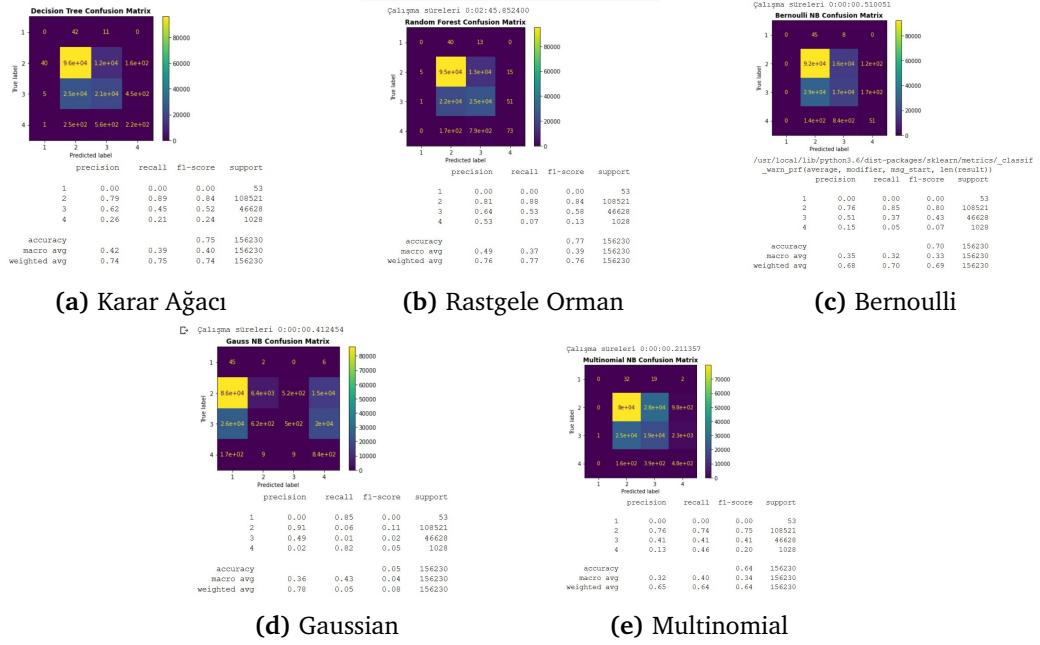
Şekil 7.1'den de inceleneyeceği üzere algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri net bir şekilde incelenebilir. Görüldüğü üzere en yüksek doğruluk payında sonuçlar üreten algoritmalar, aynı zamanda sınıflandırma algoritmalarının da temelinde bulunan Random Forest ve Decision Tree algoritmalarıdır. İlerleyen bölümlerde bu algoritmalar üzerine yoğunlaşılmış ve en yüksek doğruluk payına erişilebilecek şekilde optimize edilmişlerdir. Aşağıda ise bu algoritmaların "Accuracy",

"f-1 score" ve "Precision" değerlerinin grafikler halinde karşılaştırılması gösterilmiştir. Ayrıca optimize algoritmalar için önemli unsurlardan biri olan hız faktörü incelemesi de görülebilmektedir.

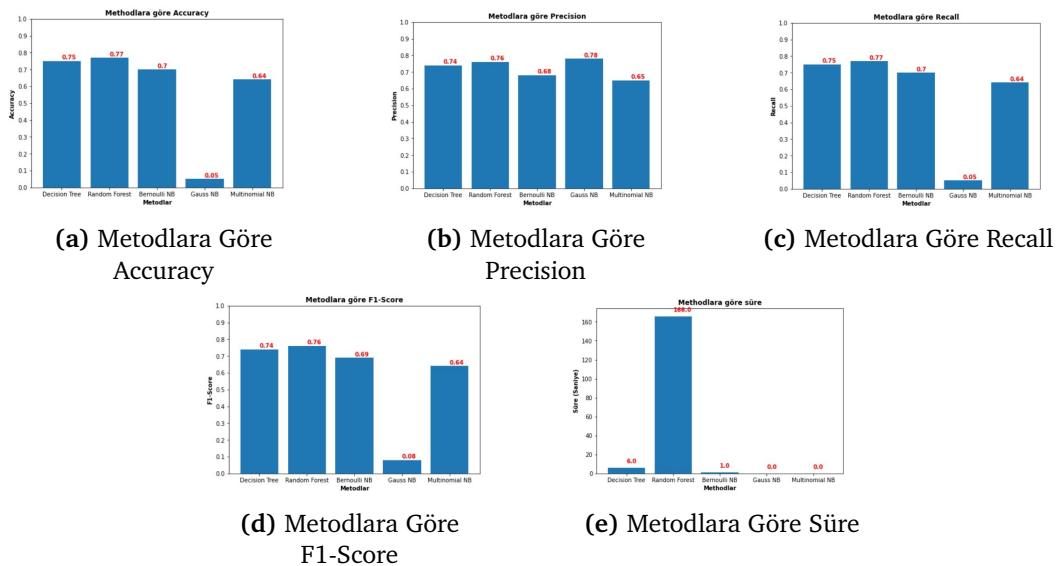


Şekil 7.2 Test boyutu = 0.1 algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri ve hız verisi

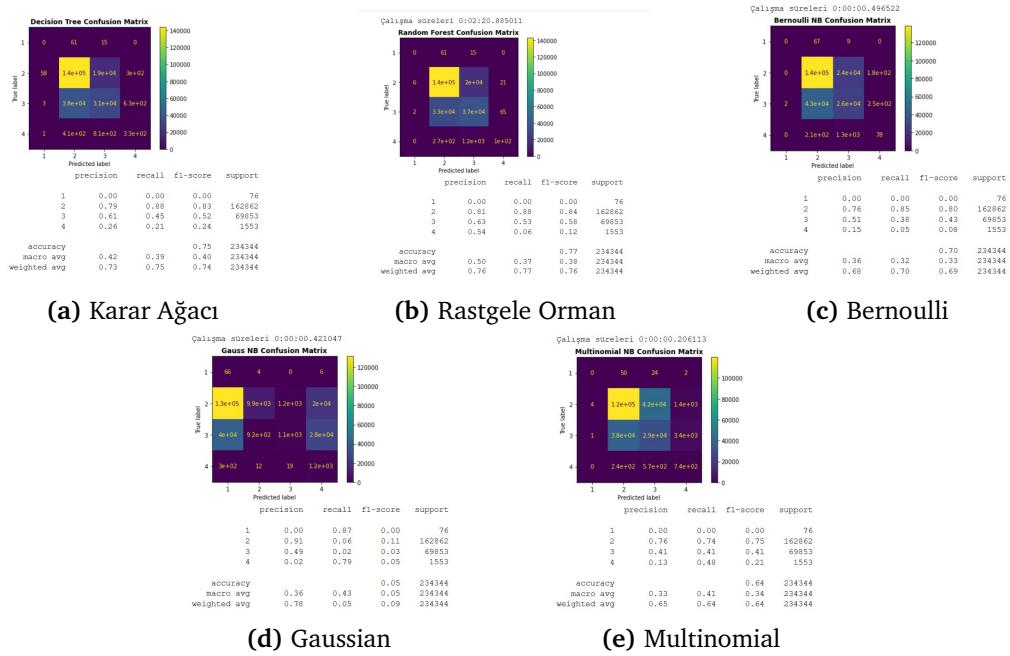
Bundan sonraki şekillerde test boyutları değiştirilerek elde edilen "Accuracy", "f-1 score" ve "Precision" değerleri, ayrıca hız faktörlerinin incelenmesi net olarak görülebilmektedir. Veriler incelendiğinde "Accuracy", "f-1 score" ve "Precision" değerleri arasında çok büyük bir farklılığın görülmediği saptanmıştır. Ayrıca DTA için "Max depth" parametresinin değişimiyle elde edilen accuracy ve çalışma süresi verileri Şekil 7.11'de gösterilmiştir. Bu verilere göre 20 ile 30 "max depth" değeri arası en optimum doğruluk değerine ulaşıldığı görülmüştür.



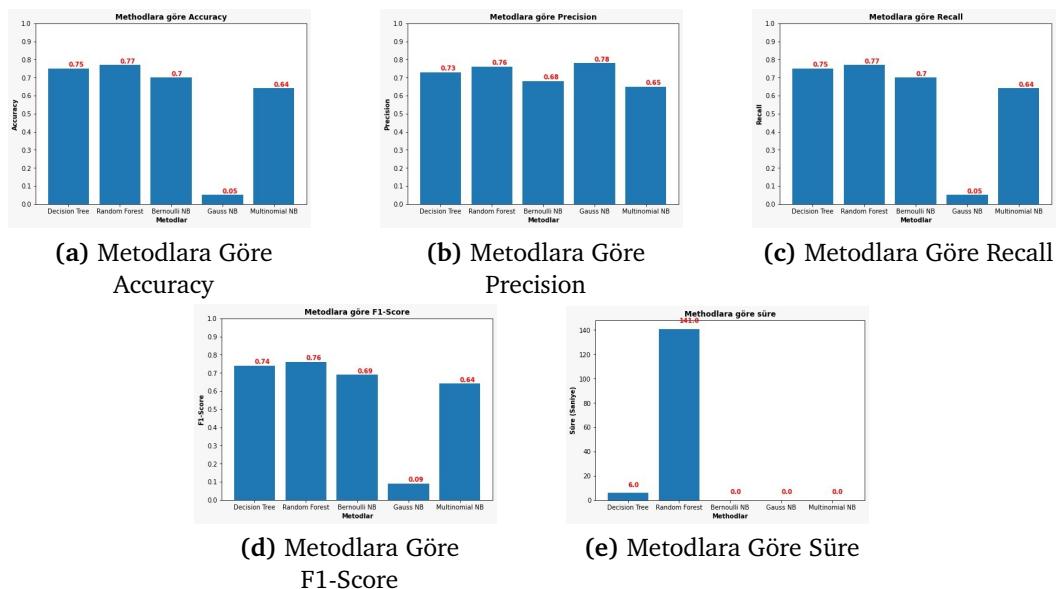
Şekil 7.3 Test boyutu = 0.2 algoritmaların karmaşıklik matrisleri (Confusion Matrix)



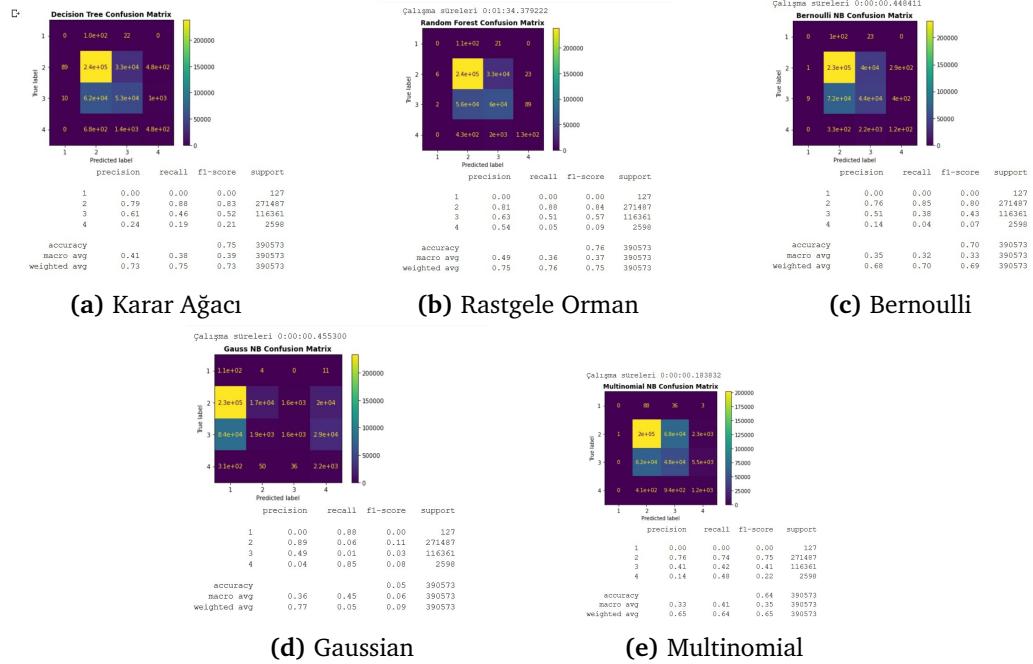
Şekil 7.4 Test boyutu = 0.2 algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri ve hız verisi



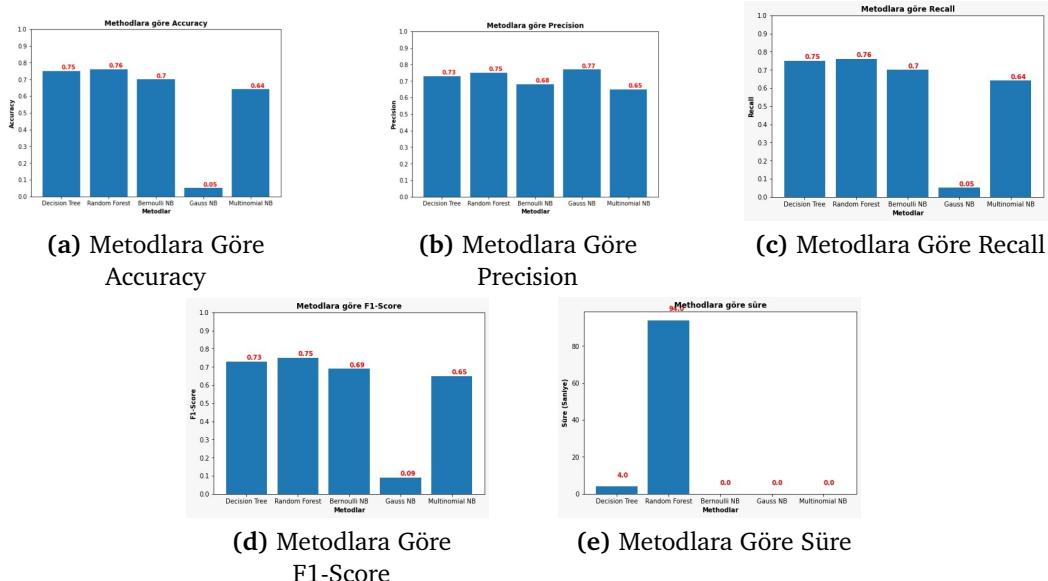
Şekil 7.5 Test boyutu = 0.3 algoritmaların karmaşıklık matrisleri (Confusion Matrix)



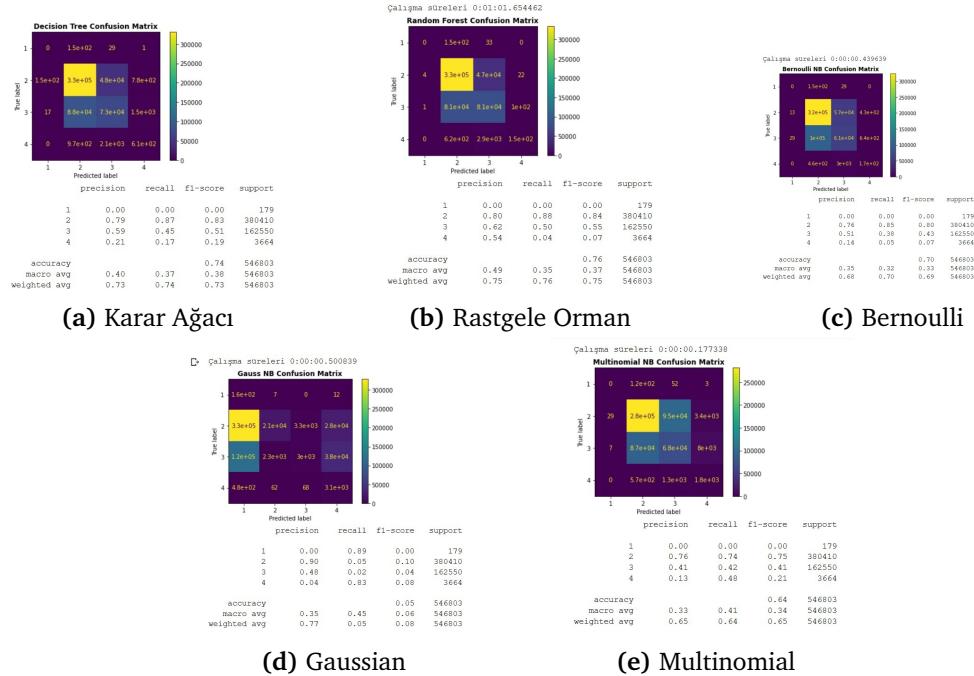
Şekil 7.6 Test boyutu = 0.3 algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri ve hız verisi



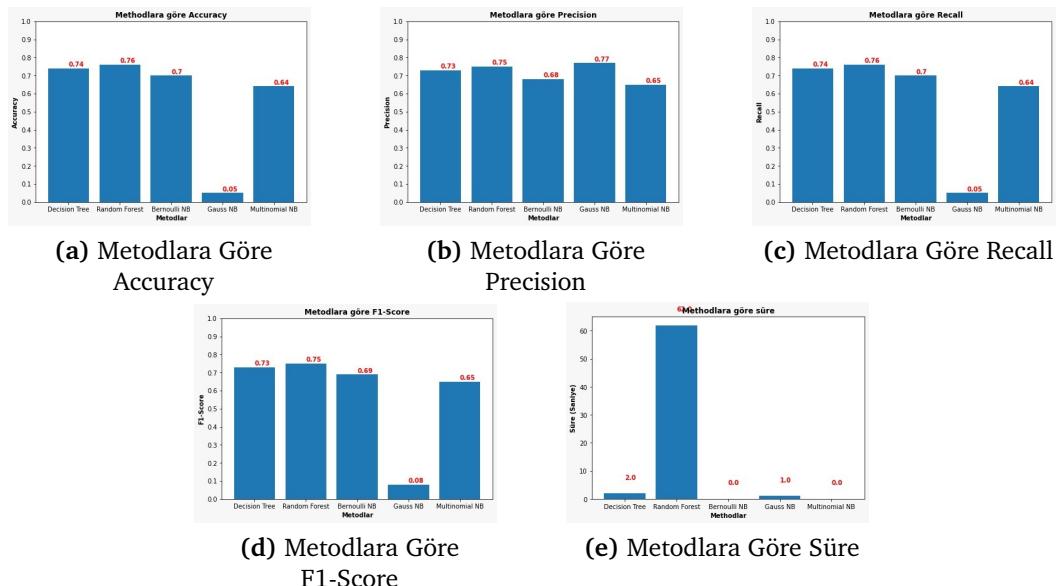
Şekil 7.7 Test boyutu = 0.5 algoritmaların karmaşıklık matrisleri (Confusion Matrix)



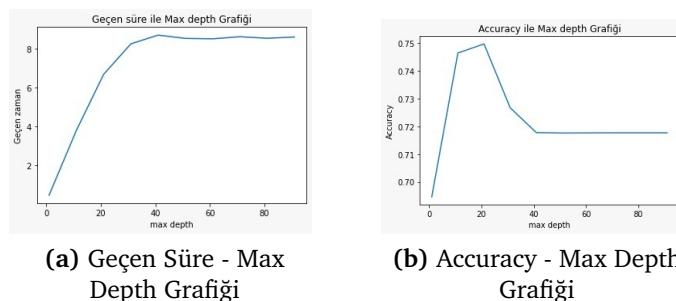
Şekil 7.8 Test boyutu = 0.5 algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri ve hız verisi



Şekil 7.9 Test boyutu = 0.7 algoritmaların karmaşıklik matrisleri (Confusion Matrix)



Şekil 7.10 Test boyutu = 0.7 algoritmaların "Accuracy", "f-1 score" ve "Precision" değerleri ve hız verisi



Şekil 7.11 Max Depth parametresinin incelenmesi

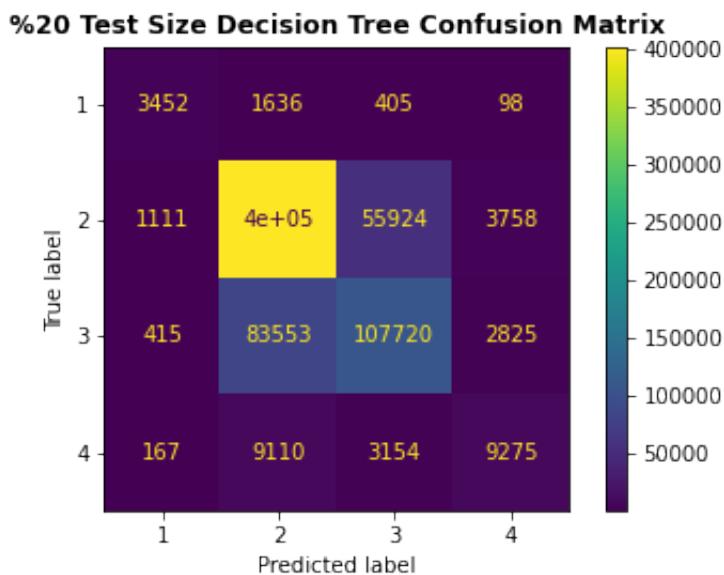
Şimdi ise az önce anlatıldığı üzere elde edilen bulguların üzerlerine optimizasyon adımları eklenerek algoritmaların doğruluk değerlerinin geliştirilmesi anlatılacaktır. Decision Tree Algoritması için en yüksek doğruluk değerine ulaşabileceğim ön görüldüğünden "max depth" değeri 0.2 olarak belirlenmiştir.

7.1 Veri Setinde Yapılan Güncellemeler

Önceki bölümde belirtilen bulgular incelediğinde trafik şiddeti olarak 1 şiddetinde olan verilerin tahmin oranının ciddi bir şekilde düşük seviyede kaldığı görülmektedir. Bu oranın 1 şiddeti için çok düşük olmasının en temel nedeni "NaN" değerlerinin 1 şiddetindeki verilerde mevcut olmasından kaynaklı olabileceği düşünülmüştür. Bunun için verisetindeki "NaN" değerlere sahip hücrelerin bir şekilde doldurulması ilk hedef olarak belirlenmiştir. Kayıp olan verilerin çoğunu nümerik değerler olduğu gözlemlendikten sonra bu değerleri doldurmak için doğruluk oranını etkileyebilecek en etkili yöntem elemesi gerçekleştirılmıştır.

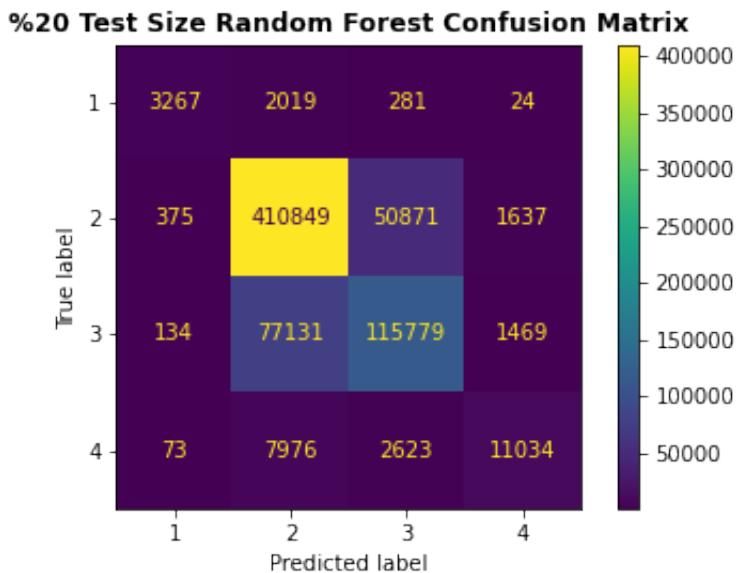
7.1.1 Ortalama Değerler

"NaN" değerlerin sütundaki diğer verilerin ortalamasının alınmasıyla doldurulduğu yöntemdir. Bu yöntem nümerik verilerden oluşan sütunlardaki verilerin doldurulmasında kullanılan en temel yöntemler arasında sayılabilir. Eksik değerler ortalamaya doldurulduktan sonra aşağıdaki bulgular elde edilmiştir. Doğruluk değerleri en yüksek olan iki algoritma incelenmiştir.



Şekil 7.12 Test boyutu = 0.2 DTA Karmaşıklık Matrisi

Şekil 7.12 'den görüldüğü üzere ilk tahminlerde 1 şiddetini tahmin etmekte zorlanan model, "NaN" değerlerin ortalaması doldurulmasıyla şiddet tahmin yeteneği olumlu yönde etkilenmiştir.

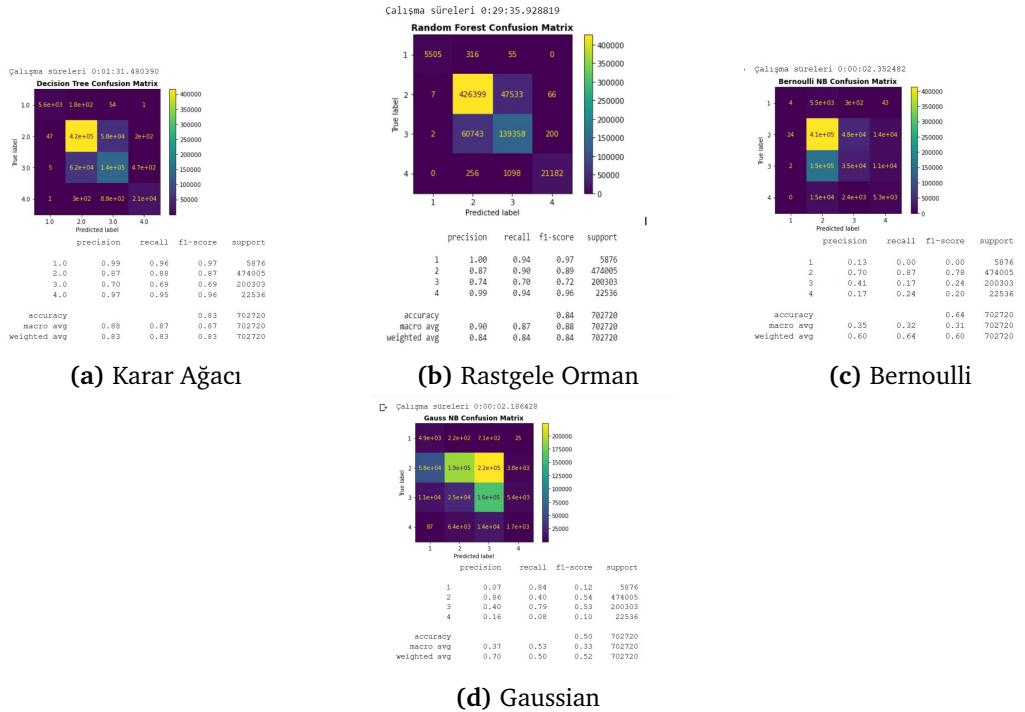


Şekil 7.13 Test boyutu = 0.1 RFA Karmaşıklık Matrisi

Şekil 7.13'de de RFA algoritmasının karmaşıklık matrisi görülmektedir. Yine önceki durumlara göre 1 kaza şiddetinin tahmin yeteneği gözle görülür bir şekilde yükselmiştir.

7.1.2 Regression Yöntemi

Regression Yöntemi, veri setinde nümerik verilerin bilinmemesi durumunda yerlerine uygun değerlerin yerleştirilmesinde oldukça popüler olarak kullanılan bir yöntemdir. Regression yöntemiyle bir kayıp değer, diğer verilerle ilişkisi tahmin ettirilerek bulunabilir. Bu modelle tanımlanan kayıp verilerin üzerine tahmin sırasında "MICE" modelinin de eklenmesiyle oldukça güçlü ve etkili bir model geliştirmemize olanak sağlamıştır. Sonuç bölümünde, Regression yöntemiyle veri setinin doldurulması ve MICE modelinin de algoritmaların modüllerine entegre edilmesiyle elde edilen sonuçlar grafikler halinde gösterilecektir. Şekil 7.14'te görüldüğü gibi Random Forest Algoritması ve Decision Tree Algoritması en yüksek verim alınan algoritmalarıdır.



Şekil 7.14 Test boyutu = 0.2 Regression Yöntemiyle doldurulan veriler ve MICE modeli entegresi sonucu oluşan her bir algoritmanın karmaşıklık matrisleri

Multinomial Naive Bayes Algoritması, MICE modeli ile entegresinde, negatif değerlerin oluşmasından dolayı düzgün çalışmamaktadır.

7.1.3 MICE Modeli ve Regression Yöntemi Entegresi

[9] MICE, kayıp verilerin, birçok defa Regression yönteminin kullanılarak doldurulduğu ve her bir kayıp verinin setteki diğer mevcut verilerle bağlılığından tahmin edildiği böylelikle veri setindeki kayıp değerlerin uygun şekilde doldurulduğu bir yöntemdir.

- İlk adımda "NaN" değerler ortalamaya doldurulur.(Herhangi başka bir basit yöntem seçilebilir.)
- İkinci adımda ortalamaya doldurulan bir veri tekrar eski haline ("NaN") döndürülür.
- Üçüncü adımda bu kayıp değer, regression yöntemi kullanılarak ve diğer verilerle ilişkisi incelenerek en uygun sonuca tahmin ettirilir.
- Dördüncü adımda elde edilen bu tahmin değeri, ikinci adımda silinen verinin yerine yazılır.

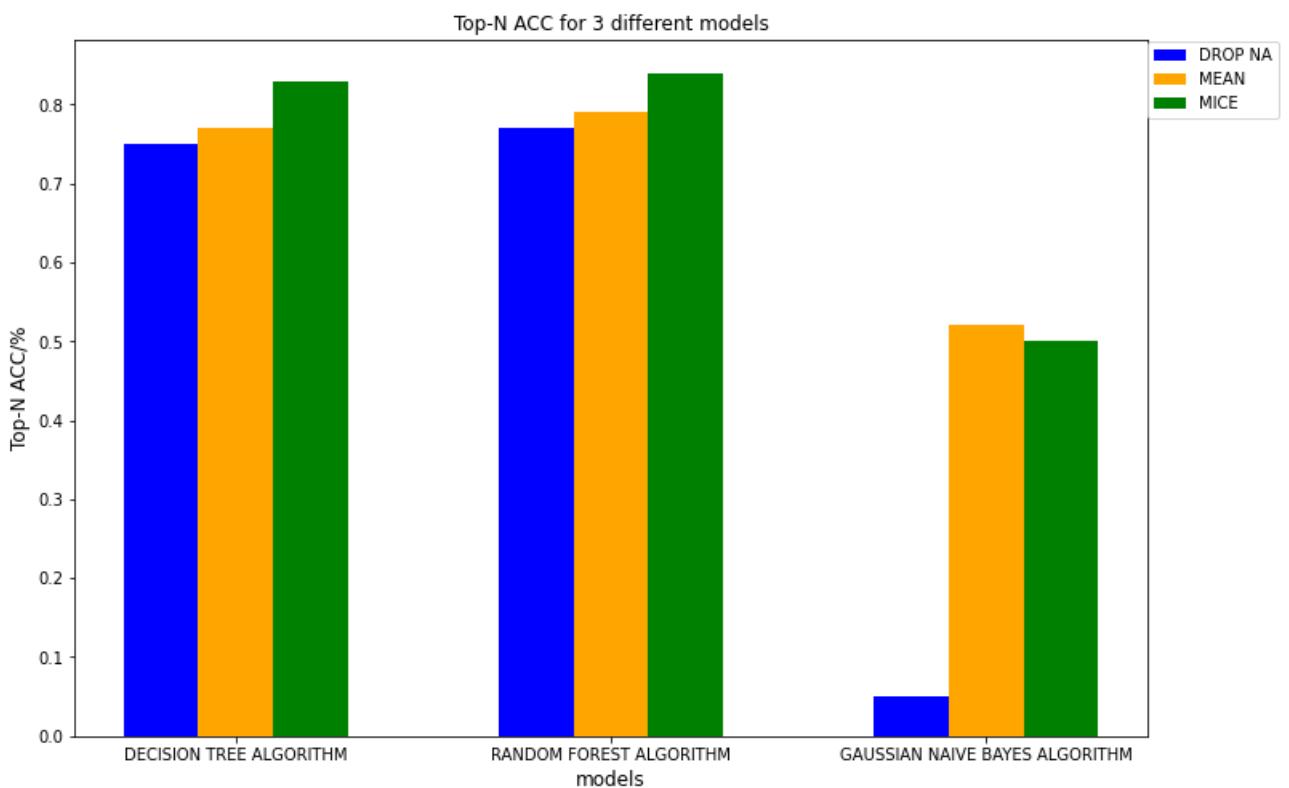
İkinci adım ve dördüncü adım her ortalamayla doldurulmuş olan veri için devam eder. Bu adımların sonunda veri setindeki kayıp veriler diğer verilere daha yakın ve modele uygun bir şekilde doldurulur.

MICE metodu modele entegre edildiğinde Şekil 8.1'de de görüldüğü gibi sınıflandırma algoritmalarının hızını ve doğruluk değerini oranını büyük ölçüde geliştirmiştir.

8 **Sonuç**

Yukarıda bahsedilen performans geliştirmeleri sonrasında gözle görülür biçimde doğruluk oranında ve sürede gelişmeler olmuştur. Özellikle Karar Ağacı (Decision Tree) Algoritmasında yüksek oranda verim alınmıştır. İlk adımlarda % 75 oranında bir tahmin yeteneği varken geliştirmelerle % 83'lere kadar yükselen bir model olmuştur. Bunlarla birlikte süre bakımından biraz zayıf kalan Rastgele Orman (Random Forest) Algoritmasının doğruluk değeri % 77'lerden % 84 oranına yaklaşmıştır. Fakat verimlilik kadar eğitimin hızı da önemli bir faktör olduğundan RFA, DTA'ya göre eğitim ve teste oldukça yavaş kaldığı görülmüştür. Bernoulli Naive Bayes Algoritması için ilk durumlarda % 70 oranlarında bir doğruluk değerine sahipken veri setindeki kayıp verilerin giderilmesindeki optimizasyonlar ile % 64 oranında doğruluk değerileyahmin yeteneği düşüş göstermiştir. Önemli ölçüde değişen bir diğer bir algoritma ise Gaussian Naive Bayes Algoritmasıdır. Veri setinin "NaN" değerlere sahip satırlarının atıldığı veri setinde % 0.05 doğruluk oranıyla düşük orandayken bu verilerin MICE yöntemiyle doldurulması sonrasında doğruluk oranı 0.5 oranlarına yükselmiştir. Bu model için düşük bir tahmin oranı olsa da gözle görülür bir artışla en fazla artışı gerçekleştirmiştir.

Geliştirilen bu model için sistemde tahmin için en uygun ve en hızlı çalışan algoritmanın Karar Ağacı Algoritması (Decision Tree Algorithm) olduğuna karar verilmiştir ve tahmin edilmesi için kullanıcılardan girilen kaza verilerinin Karar Ağacı Algoritması kullanılarak olası kaza şiddetinin tahmin edilmesi kararlaştırılmıştır.



Şekil 8.1 Sonuç Grafiği: En yüksek 3 algoritmanın değerleri

Referanslar

- [1] Z. Ronghui, “Ml to predict accident severity pa mont,” 26, 2019.
- [2] S. Moosavi, “Us accidents (3.5 million records) a countrywide traffic accident dataset (2016 - 2020),” vol. 26, 2020.
- [3] A. Navlani, “Decision tree classification in python,” 26, 2018.
- [4] A. Navlani, “Understanding random forests classifiers in python,” vol. 26, 2018.
- [5] S. Ray, “6 easy steps to learn naive bayes algorithm with codes in python and r,” vol. 26, 2017.
- [6] P. Majumder, “Gaussian naive bayes,” *Electrical Engineering at Institute of Engineering and Management (IEM), Kolkata*, vol. 26, 2020.
- [7] N. Mutha, “Bernoulli naive bayes,” vol. 26, 2020.
- [8] G. Solanki, “Multinomial naive bayes,” *Great Learning*, vol. 26, 2020.
- [9] A. Bilogur, “Mice,” *Simple Techniques for missing data imputation*, vol. 26, 2018.

Özgeçmiş

BİRİNCİ ÜYE

İsim-Soyisim: Tarık Can ŞAHİN

Doğum Tarihi ve Yeri: 27.09.1999, İstanbul

E-mail: l1117090@std.yildiz.edu.tr

Telefon: +90 537 980 22 04

Staj Tecrübeleri: Robotic Process Automation (RPA) Yazılım Geliştirme Stajyeri
AKTEK

İKİNCİ ÜYE

İsim-Soyisim: İlker BEDİR

Doğum Tarihi ve Yeri: 21.01.1998, İstanbul

E-mail: l1116036@std.yildiz.edu.tr

Telefon: +90 553 073 25 34

Staj Tecrübeleri: Kafein Yazılım Web Developer

Proje Sistem Bilgileri

Sistem ve Yazılım: Windows İşletim Sistemi, Python

Gerekli RAM: 2GB

Gerekli Disk: 256MB