Yıldız Teknik Üniversitesi
Elektrik-Elektronik Fakültesi
Bilgisayar Mühendisliği Bölümü

BLM3510 – Yapay Zeka
Prof. Dr. Mehmet Fatih Amasyalı
Proje Raporu

İsim: Ömer Diner
No: 20011017
E-Posta: omer.diner@std.yildiz.edu.tr

İsim: Selahattin Yasin Çaycı
No: 2011099
E-Posta: yasin.cayci@std.yildiz.edu.tr

Video linki: https://www.youtube.com/watch?v=MVDPVj59lWQ

Proje Konusu

Yapay zeka projesi konusu olarak "Sentetik Veri ve Gerçek Veri ile Türetilmiş Şarkı Sözlerinden Müzik Türlerini Bulma" konusunu önerdik. Projemizin amacı, şarkı sözlerinin türe göre yoğunluğunu belirlemek ve bu analizi yaparken doğal dil işleme tekniklerini kullanarak şarkı türlerini tespit etmektir.

Projenin genel olarak geliştirilme sürecini şöyle planladık:

Proje kapsamında pop, metal, rock, rap ve country türlerinden 100'er adet şarkı sözü chatgpt'ye ürettirilerek 500 adet şarkı sözünden bir veri seti oluşturulacak. Ayrıca aşağıdaki linkte yer alan müzik sözleri ve türleri veri setinden aynı miktarda veri çekilecek. Daha sonra ise bu iki veri seti eşit oranda karıştırılarak üçüncü bir veri seti elde edilecek. Bu üç veri seti de %80 train ve %20 test şeklinde ikiye ayrılacak. Scikit-Learn kütüphanesi kullanılarak Lojistik Regresyon, SVM, Multilayer Perceptron, K-Neighbors ve Random Forests ile eğitim verisi kullanılarak 5 farklı ML modeli üretilecek. K-katlı çarpraz doğrulama kullanılarak parametreler optimize edilecek ve modellerin performansları iyileştirilmeye çalışılacak.

Üretilen modeller üzerinden test veri setleri kullanılarak bir performans analizi yapılacak. Değerlendirme metrikleri accuracy, precision, recall ve f1 skoru olacak. Bu metriklere göre 3 farklı veri seti ve 5 farklı model karşılaştırılacak, hangi veri seti ve modelle en yüksek performansın elde edileceği araştırılacak.

Fatih Hoca'nın bu öneriye geribildirimi ise şöyle oldu:

"Test kümesi sadece gerçek verilerden oluşsun.

Eğitim kümesindeki sentetik veri oranının test performansına etkisi incelensin.

test 250 gerçek

eğitim 500 gerçek + 100 sentetik

eğitim 500 gerçek + 200 sentetik

eğitim 500 gerçek + 500 sentetik

eğitim 500 gerçek + 1000 sentetik "

Biz de projeyi bu doğrultuda gerçekleştirdik.

Veri Seti Hazırlama

Gerçek Veri

Kaggledan araştırma yapılarak, gerçek veriler konumuza en uygun olan veri setinden alınmıştır.

Veri setinin linki: https://www.kaggle.com/datasets/d3stron/english-music-lyrics-5-genres-500k

Bu veri seti, iki sütundan oluşmaktadır ve 5 farklı türe ait şarkıların şarkı sözleri ve tür bilgilerini içermektedir. Toplamda yaklaşık 500.000 satır bulunmaktadır. Hocanın önerdiği eğitim ve test veri miktarlarına (eğitim için 500 gerçek, test için 250 gerçek veri) uyabilmek adına bu veri setinin işlenmesi ve kırpılması gerekmektedir. Ayrıca, şarkı sözleri dışındaki gereksiz kısımların temizlenmesi amacıyla veri, Python kullanılarak işlenmiştir. Veri, bir data frame içine alınarak sütun adlandırmaları düzenlenmiş ve türlere göre sıralanmıştır. Test için her türe eşit dağılmış 250 gerçek veri içeren kaggleset250.xlsx dosyası, eğitim için ise 500 gerçek veri içeren kaggleset500.xlsx dosyası oluşturulmuştur. Bu işlemleri gerçekleştiren kod, **format_kaggleset.py** adlı dosyada yer almaktadır.

Sentetik Data

Sentetik data üretmek içinse api yardımıyla gemini pro modeline her tür için farklı promptlar yollanılarak toplam 1000 adet sentetik şarkı sözü ürettirildi.

Kullandığımız promptlara örnek vermek amaçlı rock türü için verdiğimiz inputlar:

#text1="generate me a rock song. make it sound like a song from the 80s. length: 200-250 characters. language: english"

#text2="Write me a rock song. Genre: rock. make me feel like i am in a rock concert. length: 150-250 character. language: english"

#text3="generate me a rock song lyrics. example artist: queen It should be in English and approximately 200 characters long."

#text4="i want a rock song. create me a lyrics. make it sound like a song from the 70s. length: 200-300 characters. language: english"

#text5="i am a rock singer. i want to sing a song.i like the beatles create me a rock song. length: 200-300 characters. language: english"

Bu şekilde her tür için uygun promptlar hazırlayarak **aiexceldoldur.py** dosyası 5 kere çalıştırıldı ve sentetik datalarla dolu olan 5 adet excel dosyası oluştu.

clean_data.py dosyası ile de bu 5 adet excel dosyası pandas yardımıyla işlenerek tek bir veri seti oluşturuldu ve aynı set üzerinde veri temizleme gerçekleştirildi. Bu setin kaydedildiği excel dosyası modellerin eğitimindeki sentetik veri ihtiyacını karşılayacaktır.

Geliştirme Süreci

Projenin geliştirme sürecinde 1 haftalık bir zaman dilimi konunun seçimi için ayrıldı. Bu süreçte Fatih Hoca ile de karşılıklı sohbetlerle öneri alındı.

Konu onaylandıktan sonra ise veri setini hazırlamak için 1 hafta ayrıldı. Kaggle üzerinden seçilen konuya en uygun setler araştırıldı ve karşılaştırma yapıldı. Projenin gereksinimlerine en uygun olanına karar verildikten sonra set indirilip üzerinde python dili kullanılarak veri hazırlama süreci gerçekleştirildi. Bu süreçte sütun adlarını düzenleme, veri sayısını kısıtlama ve gereksiz noktalama işaretlerini kaldırma ve veriyi türlere göre sıralama işlemi yapıldı. Bu sürecin sonunda 250 ve 500 satırlık iki adet gerçek veri excel dosyasına ulaşıldı.

Bundan sonraki süreçte sentetik veri üretmek için genel kullanıma açık en uygun yapay zeka modelinin gemini-pro olduğuna karar verildi ve bu modelin api kullanımı incelendi. 5 farklı türde her tür için farklı promptlar kullanılarak toplam 1000 adet sentetik veri üretildi. Bu veriler de temizleme işleminden geçirildikten sonra sentetik set hazır hale geldi.

Bundan sonraki süreç ise modellerin eğitimi ve geliştirilmesi süreciydi. Bu süreç de yaklaşık 2 hafta sürdü. 6 adet model ve 3 adet embedding seçimi yapıldı. Modeller çeşitli parametrelerle denendi ve veri setine en uygun olanları seçildi. İlk başlarda 0.3 olarak gözlemlenen accuracy ve f1 skorları çok sayıda yapılan çeşitli denemeler sonucunda 0.5'leri görür hale geldi. Performansı ölçmek için gözlemlenmesi gereken parametreler belirlendi ve her sonuç için kaydedildi.

Modellerin çıktıları görselleştirildi ve üzerinden yorum yapılabilecek bir çıktıya sahip olundu.

Projede kullanılan modellerin kısa açıklamaları:

Logistic Regression, sınıflandırma problemlerinde yaygın olarak kullanılan bir yöntemdir. İkili sınıflandırmada, bir olayın gerçekleşme olasılığını tahmin eder. Doğrusal bir karar sınırı oluşturur ve bağımlı değişkenin log-odds'ını bağımsız değişkenlerle ilişkilendirir. Sağlık, pazarlama ve finans gibi alanlarda yaygın olarak kullanılır.

SVM, veri noktalarını farklı sınıflara ayıran en iyi hiper düzlemi bulmaya çalışır. Veriyi sınıflandırmak için karar sınırları kullanır ve genellikle yüksek boyutlu verilerde etkilidir. Özellikle yüz tanıma ve metin sınıflandırma gibi görevlerde kullanılır

MLP, yapay sinir ağlarının bir türüdür ve bir veya daha fazla gizli katman içerir. Veri setlerinde doğrusal olmayan ilişkileri öğrenme kabiliyetine sahiptir. Derin öğrenme uygulamalarında temel yapı taşıdır ve görüntü tanıma, dil işleme gibi birçok alanda kullanılır. Eğitimi genellikle geri yayılım algoritması ile yapılır.

K-Neighbors, sınıflandırma ve regresyon problemlerinde kullanılan basit ama etkili bir yöntemdir. Bir veri noktasının sınıfını, en yakın k komşusunun sınıfına bakarak tahmin eder. Hesaplama açısından maliyetlidir ve büyük veri setlerinde yavaş olabilir. Ancak, veri dağılımı hakkında ön bilgi gerektirmediği için esnek ve kolay uygulanabilir bir yöntemdir.

Random Forests, birden fazla karar ağacının birlikte kullanılmasıyla oluşturulan bir makine öğrenme yöntemidir. Her bir ağaç, rastgele seçilmiş özellikler ve örneklerle eğitilir. Bu yöntem, aşırı uyumu azaltır ve genellikle daha yüksek doğruluk sağlar. Finans, biyoinformatik ve pazarlama gibi alanlarda kullanılır.

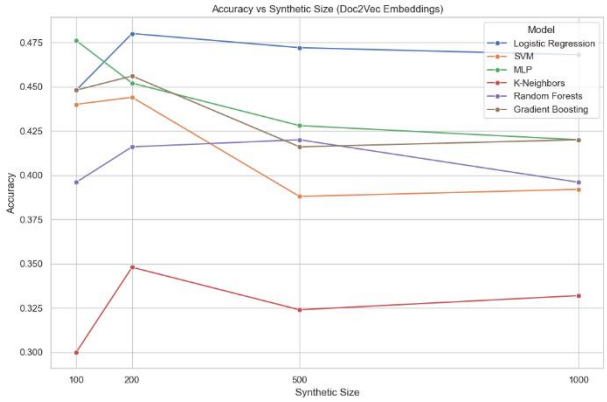
Gradient Boosting, tahmin hatalarını minimize etmek için ardışık olarak zayıf modeller oluşturur. Her yeni model, önceki modelin hatalarını düzeltmeye çalışır. Yüksek doğruluk sağlayabilen güçlü bir modeldir, ancak aşırı uyum riskine karşı dikkatli olunmalıdır. Müşteri davranışı tahmini ve kredi risk analizi gibi alanlarda yaygın olarak kullanılır.

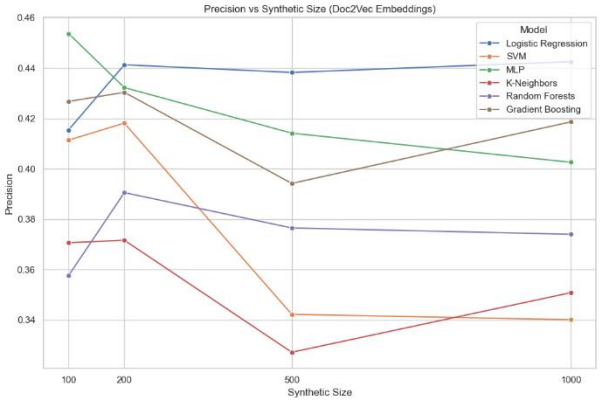
Kodun Çalıştırılması Sonucu Oluşan Çıktılar

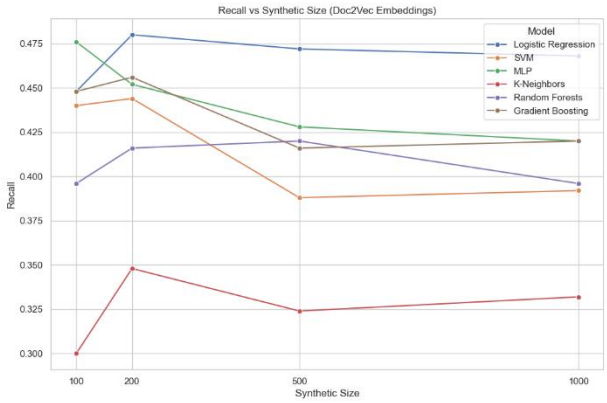
Doc2vec embeddingi ile

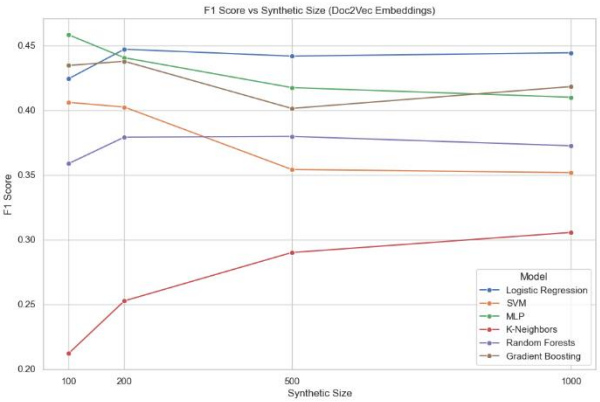
Out[31]:

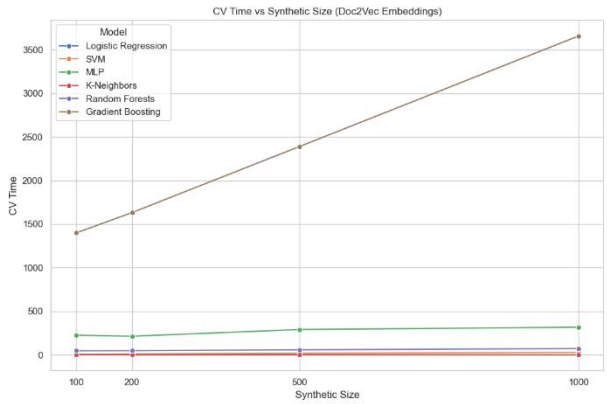
	Model	Synthetic Size	Best Params	Accuracy	Precision	Recall	F1 Score	CV Time
0	Logistic Regression	100	{'C': 0.1}	0.448	0.415421	0.448	0.424239	3.259734
1	SVM	100	{'C': 1, 'kernel': 'rbf'}	0.440	0.411371	0.440	0.405994	5.690420
2	MLP	100	{'activation': 'relu', 'alpha': 0.5, 'hidden_l...	0.476	0.453684	0.476	0.458213	224.907764
3	K-Neighbors	100	{'n_neighbors': 18}	0.300	0.370675	0.300	0.211892	0.607139
4	Random Forests	100	{'max_features': 3, 'min_samples_split': 5, 'n...	0.396	0.357605	0.396	0.358633	46.363482
5	Gradient Boosting	100	{'learning_rate': 0.1, 'loss': 'log_loss', 'ma...	0.448	0.426725	0.448	0.434504	1398.720954
6	Logistic Regression	200	{'C': 0.1}	0.480	0.441279	0.480	0.447034	0.344068
7	SVM	200	{'C': 1, 'kernel': 'rbf'}	0.444	0.418118	0.444	0.402346	9.622223
8	MLP	200	{'activation': 'relu', 'alpha': 0.001, 'hidden...	0.452	0.432232	0.452	0.440486	213.011076
9	K-Neighbors	200	{'n_neighbors': 27}	0.348	0.371655	0.348	0.252570	0.352068
10	Random Forests	200	{'max_features': 3, 'min_samples_split': 2, 'n...	0.416	0.390554	0.416	0.379106	47.496740
11	Gradient Boosting	200	{'learning_rate': 0.1, 'loss': 'log_loss', 'ma...	0.456	0.430345	0.456	0.437737	1632.368454
12	Logistic Regression	500	{'C': 0.1}	0.472	0.438240	0.472	0.441731	0.340077
13	SVM	500	{'C': 2, 'kernel': 'rbf'}	0.388	0.342259	0.388	0.354114	16.462715
14	MLP	500	{'activation': 'relu', 'alpha': 0.05, 'hidden_...	0.428	0.414146	0.428	0.417387	290.293519
15	K-Neighbors	500	{'n_neighbors': 4}	0.324	0.327212	0.324	0.289933	0.398089
16	Random Forests	500	{'max_features': 2, 'min_samples_split': 2, 'n...	0.420	0.376536	0.420	0.379700	57.247943
17	Gradient Boosting	500	{'learning_rate': 0.1, 'loss': 'log_loss', 'ma...	0.416	0.394204	0.416	0.401297	2390.181488
18	Logistic Regression	1000	{'C': 0.1}	0.468	0.442508	0.468	0.444299	0.418084
19	SVM	1000	{'C': 1, 'kernel': 'rbf'}	0.392	0.340136	0.392	0.351694	27.867291
20	MLP	1000	{'activation': 'relu', 'alpha': 0.1, 'hidden_l...	0.420	0.402592	0.420	0.409948	316.799491
21	K-Neighbors	1000	{'n_neighbors': 5}	0.332	0.350849	0.332	0.305448	0.455102
22	Random Forests	1000	{'max_features': 3, 'min_samples_split': 2, 'n...	0.396	0.374017	0.396	0.372445	71.518401
23	Gradient Boosting	1000	{'learning_rate': 0.1, 'loss': 'log_loss', 'ma...	0.420	0.418603	0.420	0.418160	3658.326730


Accuracy vs Synthetic Size (Doc2Vec Embeddings)
Accuracy
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting


Precision vs Synthetic Size (Doc2Vec Embeddings)
Precision
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting


Recall vs Synthetic Size (Doc2Vec Embeddings)
Recall
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting

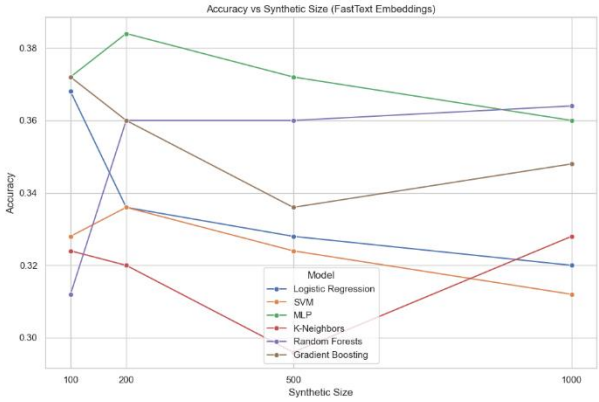

F1 Score vs Synthetic Size (Doc2Vec Embeddings)
F1 Score
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting

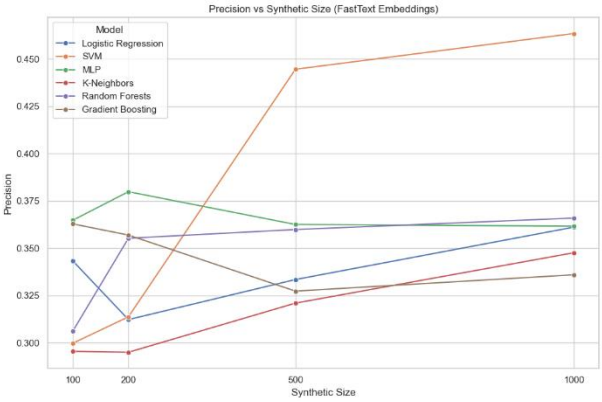

CV Time vs Synthetic Size (Doc2Vec Embeddings)
CV Time
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting

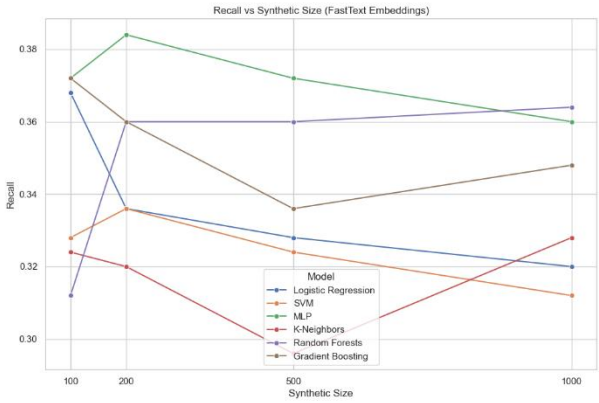
FastText embeddingi ile

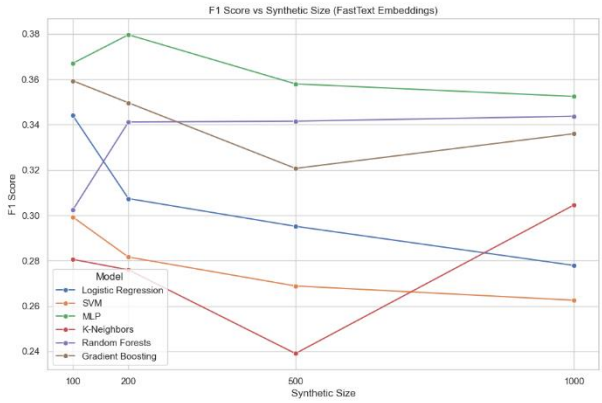
Out[32]:

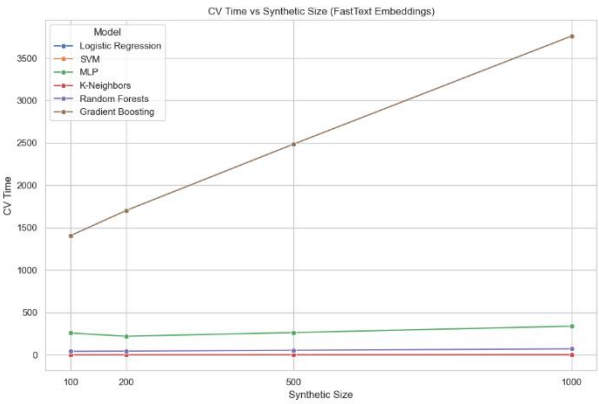
	Model	Synthetic Size	Best Params	Accuracy	Precision	Recall	F1 Score	CV Time
0	Logistic Regression	100	{'C': 10}	0.368	0.343310	0.368	0.344069	0.301048
1	SVM	100	{'C': 10, 'kernel': 'linear'}	0.328	0.299764	0.328	0.299276	0.743188
2	MLP	100	{'activation': 'relu', 'alpha': 0.0001, 'hidde...	0.372	0.364750	0.372	0.367055	256.748871
3	K-Neighbors	100	{'n_neighbors': 21}	0.324	0.295499	0.324	0.280582	0.342075
4	Random Forests	100	{'max_features': 2, 'min_samples_split': 2, 'n...	0.312	0.306139	0.312	0.302323	41.248330
5	Gradient Boosting	100	{'learning_rate': 0.1, 'loss': 'log_loss', 'ma...	0.372	0.362843	0.372	0.359373	1405.713256
6	Logistic Regression	200	{'C': 10}	0.336	0.312286	0.336	0.307403	0.330074
7	SVM	200	{'C': 10, 'kernel': 'linear'}	0.336	0.313742	0.336	0.281609	0.942211
8	MLP	200	{'activation': 'tanh', 'alpha': 0.01, 'hidden_...	0.384	0.379829	0.384	0.379651	219.174469
9	K-Neighbors	200	{'n_neighbors': 21}	0.320	0.295029	0.320	0.275975	0.440101
10	Random Forests	200	{'max_features': 3, 'min_samples_split': 10, '...	0.360	0.355249	0.360	0.341167	44.084013
11	Gradient Boosting	200	{'learning_rate': 0.1, 'loss': 'log_loss', 'ma...	0.360	0.356929	0.360	0.349661	1702.403240
12	Logistic Regression	500	{'C': 10}	0.328	0.333446	0.328	0.295178	0.414073
13	SVM	500	{'C': 9, 'kernel': 'linear'}	0.324	0.444521	0.324	0.268857	1.687366
14	MLP	500	{'activation': 'relu', 'alpha': 0.5, 'hidden_l...	0.372	0.362536	0.372	0.357987	262.352576
15	K-Neighbors	500	{'n_neighbors': 40}	0.296	0.320981	0.296	0.239057	0.372065
16	Random Forests	500	{'max_features': 3, 'min_samples_split': 5, 'n...	0.360	0.359854	0.360	0.341538	53.151015
17	Gradient Boosting	500	{'learning_rate': 0.1, 'loss': 'log_loss', 'ma...	0.336	0.327323	0.336	0.320708	2483.955027
18	Logistic Regression	1000	{'C': 10}	0.320	0.361123	0.320	0.277920	0.569109
19	SVM	1000	{'C': 10, 'kernel': 'linear'}	0.312	0.463403	0.312	0.262573	3.319749
20	MLP	1000	{'activation': 'tanh', 'alpha': 0.05, 'hidden_...	0.360	0.361613	0.360	0.352450	337.968509
21	K-Neighbors	1000	{'n_neighbors': 14}	0.328	0.347596	0.328	0.304556	0.450870
22	Random Forests	1000	{'max_features': 3, 'min_samples_split': 2, 'n...	0.364	0.365909	0.364	0.343723	70.392939
23	Gradient Boosting	1000	{'learning_rate': 0.1, 'loss': 'log_loss', 'ma...	0.348	0.335955	0.348	0.336012	3760.266709


Accuracy vs Synthetic Size (FastText Embeddings)
Accuracy
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting


Precision vs Synthetic Size (FastText Embeddings)
Precision
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting


Recall vs Synthetic Size (FastText Embeddings)
Recall
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting

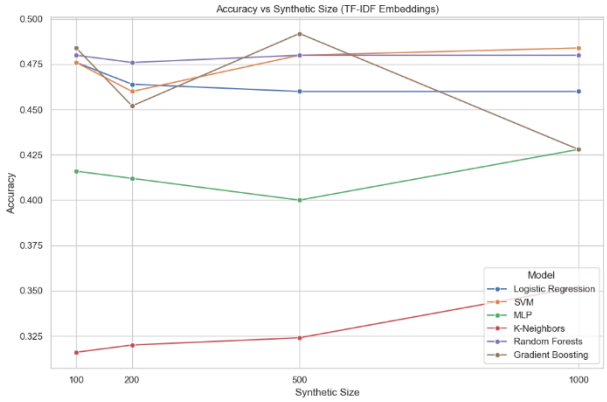

F1 Score vs Synthetic Size (FastText Embeddings)
F1 Score
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting

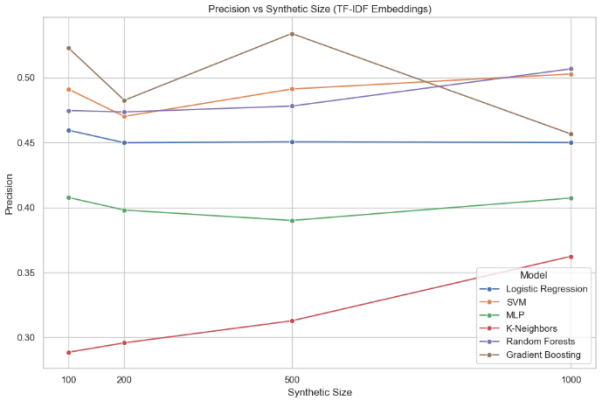

CV Time vs Synthetic Size (FastText Embeddings)
CV Time
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting

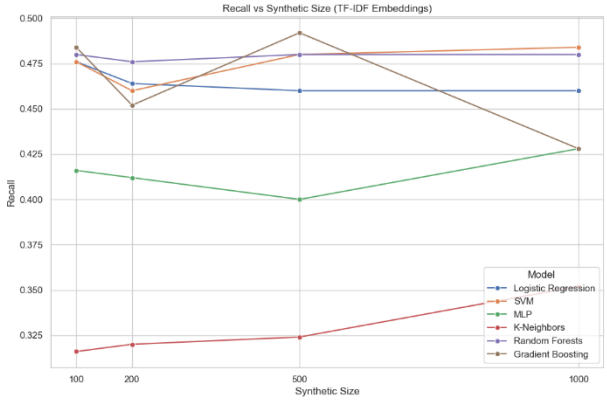
TF-IDF embeddingi ile

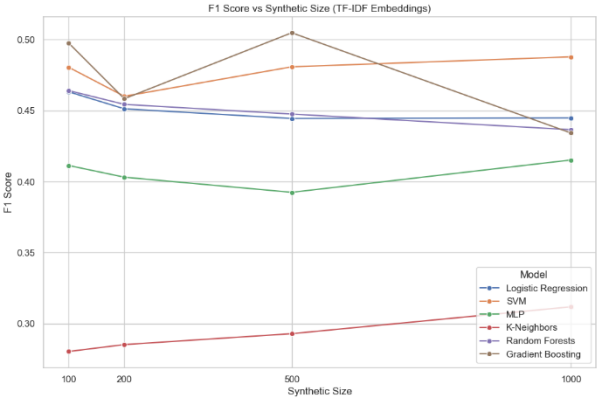
Out[24]:

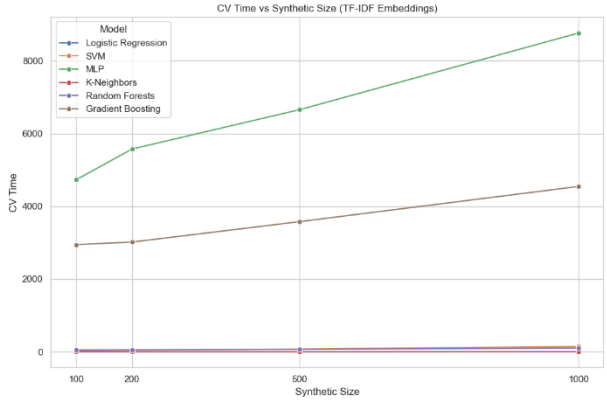
	Model	Synthetic Size	Best Params	Accuracy	Precision	Recall	F1 Score	CV Time
0	Logistic Regression	100	{'C': 1}	0.476	0.459483	0.476	0.463301	4.410167
1	SVM	100	{'C': 2, 'kernel': 'rbf'}	0.476	0.491017	0.476	0.480617	37.240186
2	MLP	100	{'activation': 'tanh', 'alpha': 0.1, 'hidden_l...	0.416	0.407729	0.416	0.411381	4731.742000
3	K-Neighbors	100	{'n_neighbors': 3}	0.316	0.288610	0.316	0.280298	0.765173
4	Random Forests	100	{'max_features': 2, 'min_samples_split': 20, '...	0.480	0.474626	0.480	0.464165	51.183553
5	Gradient Boosting	100	{'learning_rate': 0.1, 'loss': 'log_loss', 'ma...	0.484	0.522761	0.484	0.497754	2946.539431
6	Logistic Regression	200	{'C': 1}	0.464	0.449880	0.464	0.451284	1.794803
7	SVM	200	{'C': 1, 'kernel': 'rbf'}	0.460	0.470143	0.460	0.460106	44.633065
8	MLP	200	{'activation': 'tanh', 'alpha': 0.05, 'hidden_...	0.412	0.398043	0.412	0.403122	5575.098336
9	K-Neighbors	200	{'n_neighbors': 3}	0.320	0.295863	0.320	0.285183	0.765165
10	Random Forests	200	{'max_features': 3, 'min_samples_split': 10, '...	0.476	0.473572	0.476	0.454415	54.769373
11	Gradient Boosting	200	{'learning_rate': 0.01, 'loss': 'log_loss', 'm...	0.452	0.482514	0.452	0.458399	3017.319192
12	Logistic Regression	500	{'C': 1}	0.460	0.450657	0.460	0.444486	1.976290
13	SVM	500	{'C': 2, 'kernel': 'rbf'}	0.480	0.491199	0.480	0.480874	76.449236
14	MLP	500	{'activation': 'tanh', 'alpha': 0.5, 'hidden_l...	0.400	0.390031	0.400	0.392445	6655.988384
15	K-Neighbors	500	{'n_neighbors': 3}	0.324	0.312800	0.324	0.292861	0.920187
16	Random Forests	500	{'max_features': 2, 'min_samples_split': 10, '...	0.480	0.478051	0.480	0.447593	66.721060
17	Gradient Boosting	500	{'learning_rate': 0.01, 'loss': 'log_loss', 'm...	0.492	0.533918	0.492	0.504857	3578.294918
18	Logistic Regression	1000	{'C': 1}	0.460	0.450044	0.460	0.444879	2.264044
19	SVM	1000	{'C': 2, 'kernel': 'rbf'}	0.484	0.502766	0.484	0.487899	146.198916
20	MLP	1000	{'activation': 'tanh', 'alpha': 0.1, 'hidden_l...	0.428	0.407299	0.428	0.415213	8763.940091
21	K-Neighbors	1000	{'n_neighbors': 6}	0.352	0.362414	0.352	0.311801	3.349137
22	Random Forests	1000	{'max_features': 2, 'min_samples_split': 20, '...	0.480	0.506716	0.480	0.436564	105.067481
23	Gradient Boosting	1000	{'learning_rate': 0.01, 'loss': 'log_loss', 'm...	0.428	0.456620	0.428	0.434296	4546.362362


Accuracy vs Synthetic Size (TF-IDF Embeddings)
Accuracy
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting


Precision vs Synthetic Size (TF-IDF Embeddings)
Precision
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting


Recall vs Synthetic Size (TF-IDF Embeddings)
Recall
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting


F1 Score vs Synthetic Size (TF-IDF Embeddings)
F1 Score
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting


CV Time vs Synthetic Size (TF-IDF Embeddings)
CV Time
Synthetic Size
Model
Logistic Regression
SVM
MLP
K-Neighbors
Random Forests
Gradient Boosting

Sistemin Sayısal Başarısı

Model Performansını Ölçmek İçin Kullanılan Metrikler

Accuracy (Doğruluk): Tüm tahminler içinde doğru tahminlerin oranı.

Precision (Kesinlik): Pozitif tahminlerin ne kadarının gerçekten pozitif olduğu.

Recall (Duyarlılık): Gerçek pozitiflerin ne kadarının doğru tahmin edildiği.

F1-Score: Precision ve recall'un dengeli bir ortalaması.

CV Time: Model optimizasyonunun başlama ve bitiş süresi arasındaki fark.

Yukarıdaki 5 metrik her sentetik data boyutu için ve her model için ölçüldü ve bir sözlükte saklandı. Tek bir metriğe dayanmaktansa 5 adet metrik seçimi ile modelin performansının daha dengeli bir şekilde ölçülmesi ve güvenilir sonuçlar alınması hedeflendi.

İlk denemelerde %20-30 arasında bir doğruluk oranı yakalarken sonrasında koda eklenen yeni embeddingler, modeller ve optimizasyon parametreleri ile bu oran %50'leri aşabildi. Daha güçlü bir bilgisayar ile parametreler artırılarak bu oranın da üstü görülebilir.

Yorumlar

- Grafiklerden de görüldüğü üzere sentetik data miktarı arttıkça bazı modellerde performans artsa da, her deneme için modelin performansı artacak diye bir genelleme yapılamaz. Yani her veri seti için geçerli olabilecek evrensel bir model olmadığı kuralının geçerliliği tekrardan kanıtlanmıştır. Farklı veri setleri için modellerin performansı değişken olabiliyor ve modeli kuran kişinin bunu göz önünde bulundurması gereklidir.

- Eğitim amaçlı kullanılan gerçek verinin 500 adetle sınırlı kalması ve kullanılan sentetik data boyutunun da yine kısıtlı sayıda olmasından kaynaklı olarak performansın bir miktar düşük çıktığı söylenebilir. Aradaki farkı daha net olarak gözlemlemek için kullanılan veri miktarları artırılabilir. Tabi bu durumda eğitim süreci için harcanacak sürenin oldukça uzayacak olması da göz önünde bulundurulmalıdır.

- TF-IDF, FastText ve Doc2Vec dışında farklı word embeddingler kullanılarak performans iyileştirilebilir.

- Kullanılan FastText ve Doc2Vec embeddingleri için hazır modeller kullanılarak performans iyileştirilebilir. Fakat işlem süresi de göz önünde bulunmalıdır.

- Genel olarak MLP ve GBM algoritmaları en iyi performansı gösterse de işlem süresi de hesaba katıldığında hem kısa sürede optimize edilebilmesi hem de en yüksek performansa yakın performans gösterebilmesi sebebiyle Random Forest algoritması bu çalışma kapsamında en başarılı algoritma seçilmiştir.

Kullanılan Kaynaklar

https://www.labelf.ai/blog/what-is-accuracy-precision-recall-and-f1-score

https://medium.com/@rdkulkarni21/simple-linear-regression-using-random-forest-svm-and-mlp-99d1e93d67f3

https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

https://scikit-learn.org/stable/modules/grid_search.html

https://www.kaggle.com/code/willkoehrsen/intro-to-model-tuning-grid-and-random-search

https://ayselaydin.medium.com/1-text-preprocessing-techniques-for-nlp-37544483c007