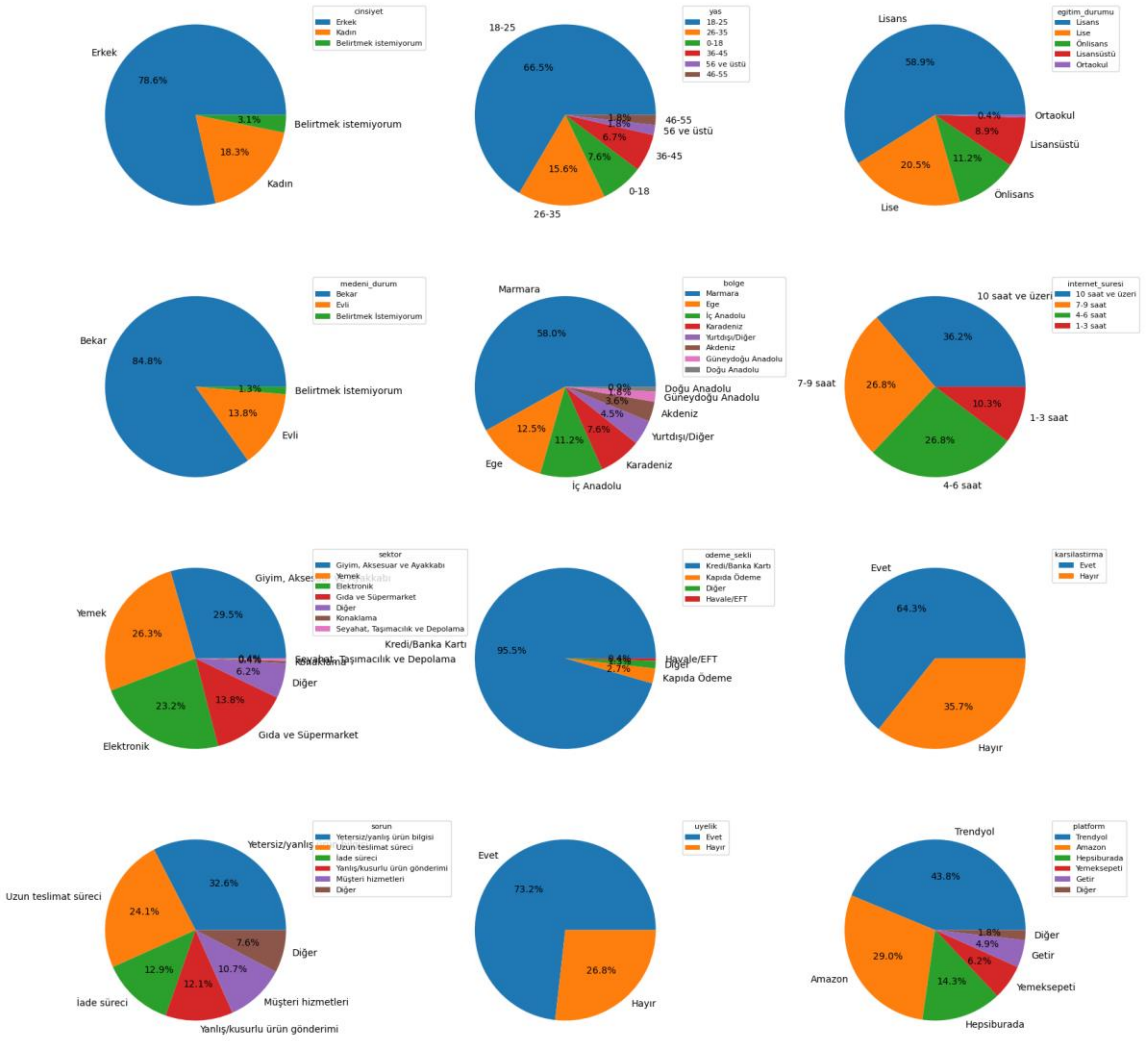
Yapay Zeka
Ödev #2 Rapor

Cihat İslam Dede – 19011047
Onur Alkan – 19011611

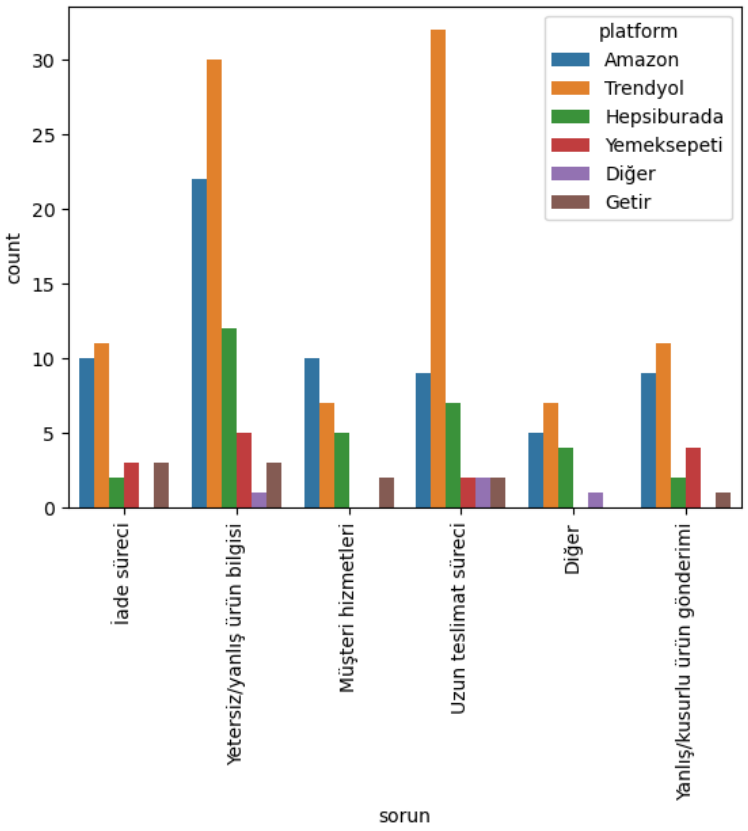
1- Veri Seti

Veri setimizi Google Form aracılığıyla, kullanıcıların E-ticaret Platformu tercihlerini belirtmeleriyle ilgili bir form üzerinden oluşturduk. Veri setimiz 12 adet özellikten oluşmakta ve kullanıcıların cinsiyet, yaş, eğitim durumu, medeni durum, yaşadıkları bölge, günlük internet kullanım süresi, en çok alışveriş yaptıkları sektör, tercih ettikleri ödeme şekli, ürün satın almadan önce karşılaştırma sitelerine bakıp bakmadıkları, e-ticaret sitelerinde en sık karşılaştıkları sorun, kullanılan e-ticaret platformu üyeliği bulunup bulunmaması ve en çok tercih ettikleri e-ticaret platformu gibi bilgiler içermektedir. *Tercih edilen e-ticaret platformu* ise modelimizin tahmin etmeye çalıştığı hedef özelliktir. Form, *224 kişi* tarafından doldurulmuştur.

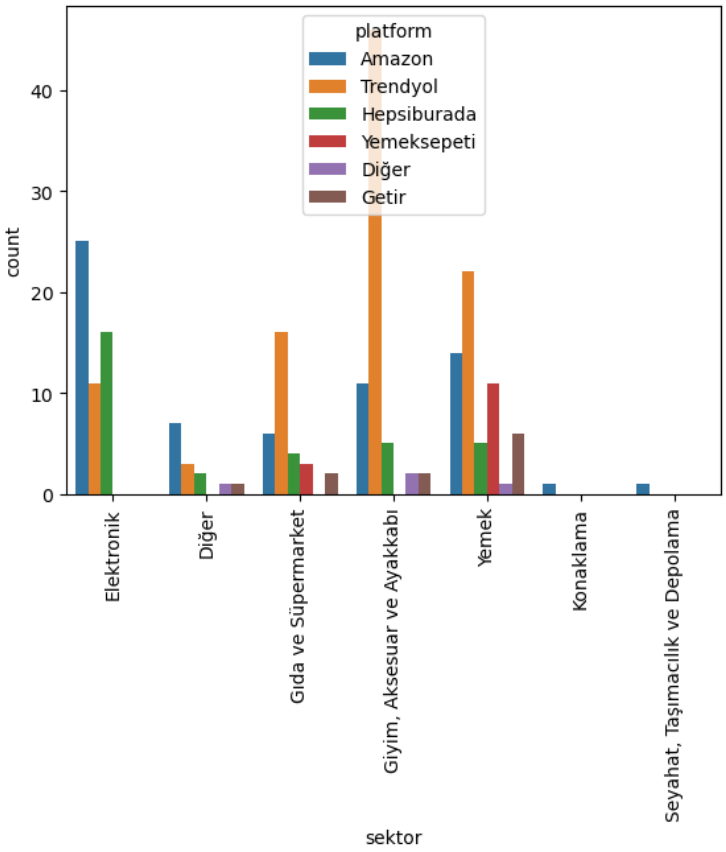
Formumuz genel olarak çevremizdeki ve kullandığımız uygulamalardaki (WhatsApp, Telegram, Discord vb.) kişiler tarafından doldurulduğu için bazı özelliklerde dağılımlar dengesiz gerçekleşmiştir. (cinsiyet, medeni durum gibi)



Şekil 1- Özelliklerin Dağılım Grafikleri



Şekil 2- Yaşanılan Sorun - Tercih Edilen Platform İlişkisi



Şekil 3- Tercih Edilen Sektör - Tercih Edilen Platform İlişkisi

2- Sınıflandırma Algoritmalarının Karşılaştırılması

7 farklı sınıflandırma algoritması kullanılarak veri seti üzerindeki ortalama accuracy performansları ölçülmüştür. *sklearn* kütüphanesinde buluna *cross_val_score()* fonksiyonu yardımıyla 10 katlı çaprazlama ile gerçekleştirilmiştir.

Kullanılan sınıflandırma algoritmaları:

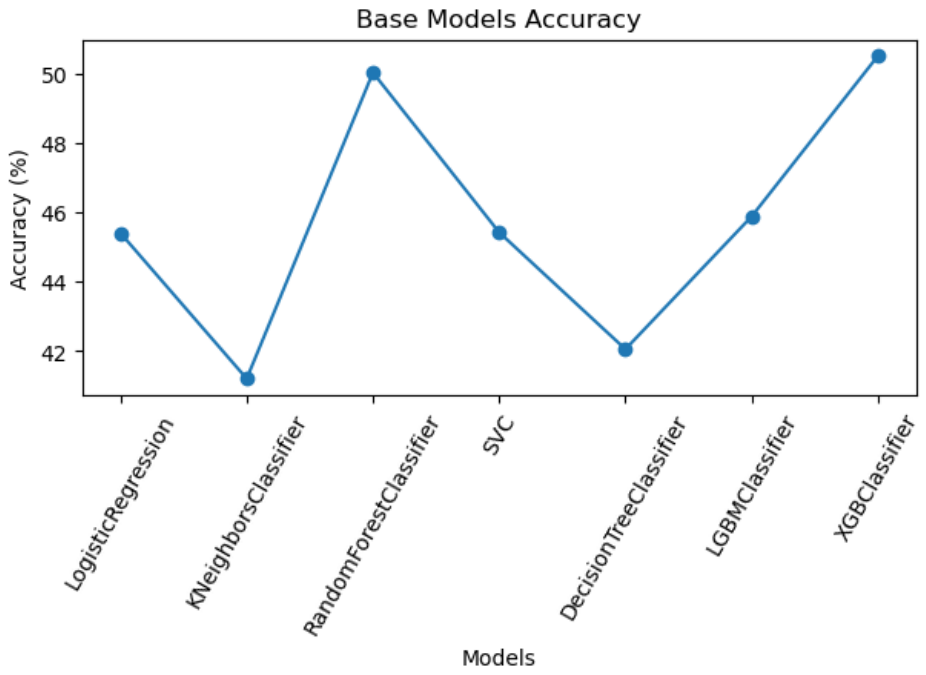
1. Logistic Regression
2. K-Nearest Neighbors (KNN)
3. Random Forest
4. Support Vector Machine (SVM)
5. Decision Tree
6. Light Gradient Boosting Machine (LightGBM)
7. eXtreme Gradient Boosting (XGBoost)

A- Varsayılan Parametreler (Baselbaselineine)

Algoritmalar öncelikle varsayılan parametreler ile çalıştırılmıştır. 10 katlı çaprazlama ile gerçekleştirileceği için *cv* (cross-validation) değeri 10 olarak atanmıştır.

```
1  lr = LogisticRegression(random_state=42)
2  lr_cv = cross_val_score(lr, X, y, cv=10)
3  lr_default = round(lr_cv.mean(), 4) * 100
4  print("LogisticRegression average performance: %", lr_default)
✓ 0.2s
LogisticRegression average performance: % 45.410000000000004
```

Şekil 4-Varsayılan Parametre ile Çalıştırılma Örneği



Şekil 5- Varsayılan Parametreler ile Performans Karşılaştırmaları

DecisionTree ve KNN, Random Forest ve XGB gibi algoritmalara göre daha kötü performans göstermiştir. Diğer algoritmalar yaklaşık olarak yakın sonuçlar oluşturmuştur.

B- Hiper-parametre Seçimi (Hyperparameter Tuning)

Daha sonra model performanslarımızı iyileştireceğini düşündüğümüz için modellerimizi belirlediğimiz bazı parametreler ile tekrardan çalıştırdık. Burada *sklearn* kütüphanesinde bulunan *GridSearchCV* fonksiyonundan yararlandık. Yaptığımız araştırmalar sonucunda *GridSearchCV* fonksiyonunun *RandomizedSearchCV* fonksiyonuna göre küçük veri setlerinde daha iyi çalıştığını öğrendik ve yaptığımız denemeler sonucunda bunu doğruladık.

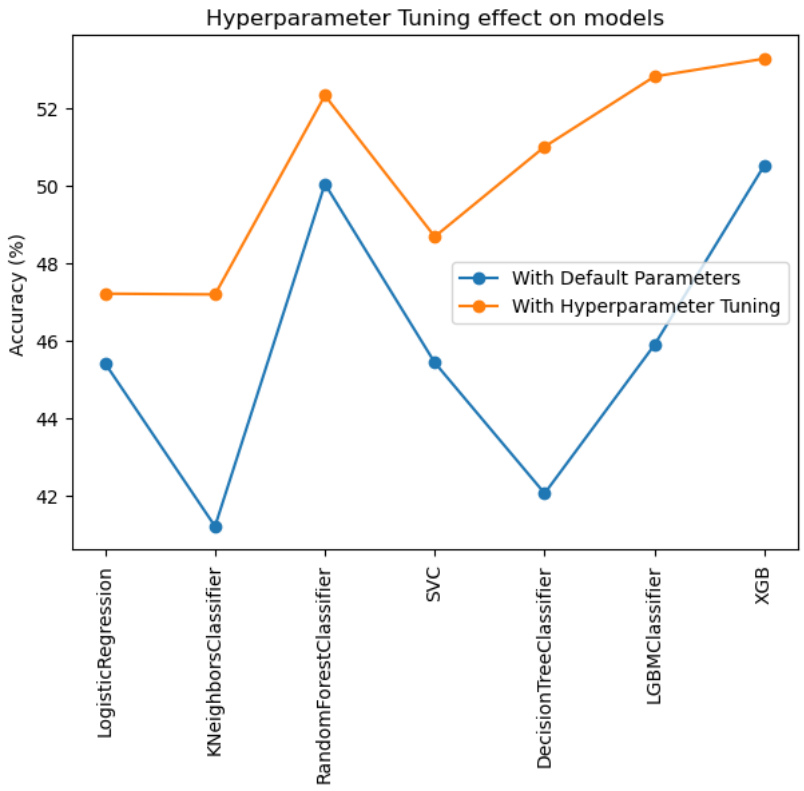
```
lr = LogisticRegression()
param_grid = {
    "max_iter": [100, 300, 500, 1000],
    "penalty": ["l1", "l2", "elasticnet", "none"],
    "C": np.logspace(-4, 4, 20, 50, 100),
    "solver": ["liblinear", "newton-cg", "lbfgs", "sag", "saga"],
}

clf_lr = GridSearchCV(lr, param_grid=param_grid, cv=10, verbose=True, n_jobs=-1)
best_clf_lr = clf_lr.fit(X, y)
clf_performance(best_clf_lr, "Logistic Regression")
```

✓ 37.9s

```
Fitting 10 folds for each of 1600 candidates, totalling 16000 fits
Logistic Regression
Best Score: %47.21
Best Parameters: {'C': 0.007847599703514606, 'max_iter': 100, 'penalty': 'l2', 'solver': 'newton-cg'}
```

Şekil 6- Hiper-parametre ile Çalıştırılma Örneği



Şekil 7-Hiper-parametre Seçimi ile Performans Karşılaştırmaları

Hiper-parametre seçimi sonrası tüm modellerin performanslarında iyileşmeler gözlemlenmiştir.

Hiper-parametre ile elde edilen modeller arasından Logistic Regression ve diğer modellerin T-test karşılaştırılması yapılmıştır.

```
T-test comparison between Logistic Regression and KNN
T-value:  -9.409735264818606
P-value:  7.878782295709842e-18
T-test comparison between Logistic Regression and SVC
T-value:  -4.579913728019071
P-value:  7.873628877269608e-06
T-test comparison between Logistic Regression and Random Forest
T-value:  -9.58500113177016
P-value:  2.408400811200256e-18
T-test comparison between Logistic Regression and Decision Tree
T-value:  -3.1823509078946572
P-value:  0.0016770595539228337
T-test comparison between Logistic Regression and LGBM
T-value:  -2.578659445541903
P-value:  0.010585809707624579
T-test comparison between Logistic Regression and XGB
T-value:  -1.9961307016989545
P-value:  0.04718137493864809
```

T-testi sonucunda ortaya çıkan p-value'yi yorumlamak için bir alpha değeri belirlememiz gerekir. Bu değeri projemiz için 0.05 kabul edersek, p-value 0.05'ten az olmadığı için sıfır hipotezini reddetmekte başarısız oluruz. Bu, iki model performansı arasında, modellerin ortalama ağırlıklarının farklı olduğunu söylemek için yeterli kanıtımız olmadığı anlamına gelir. Sonuç olarak, p-value KNN, SVC ve Random Forest modelleri için 0.05 değerinden büyüktür ve anlamlı farklı olduğunu söylemek için bir kanıtımız yoktur.

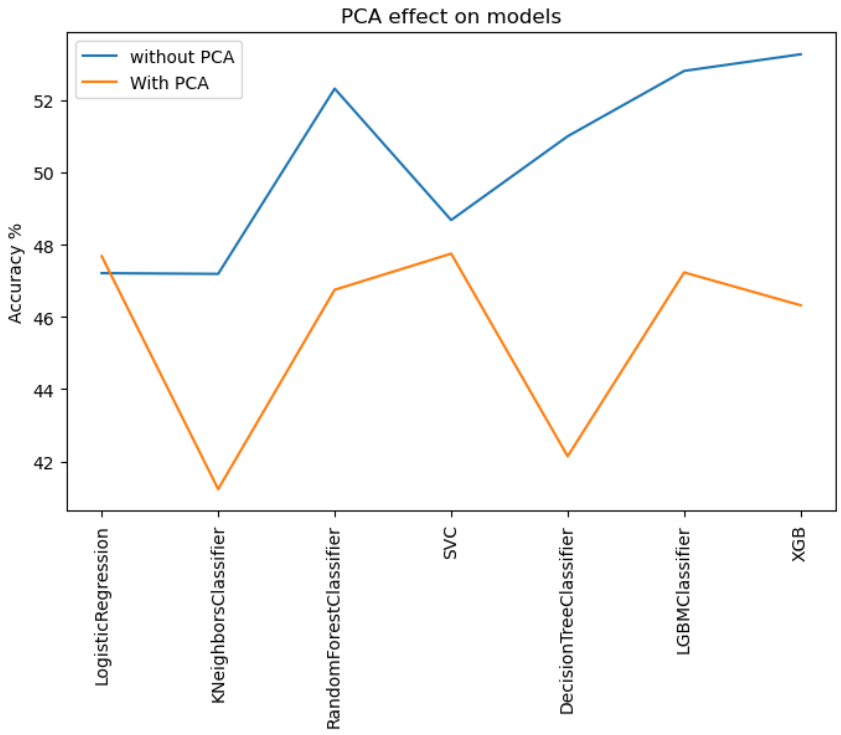
C- Özellik Dönüşümü (PCA = Principal Component Analysis)

sklearn kütüphanesinde bulunan *PCA* fonksiyonu ile özellik dönüşümü yapılmıştır. Dönüşüm sonrası özellik sayısı 7'ye düşmüştür.

```
1  from sklearn.decomposition import PCA
2
3  pca = PCA(n_components=0.95, random_state=42)
4  X_pca = pca.fit_transform(X)
5  print(X_pca.shape)
✓ 0.0s
(216, 7)
```

Şekil 8- PCA Dönüşümü

PCA dönüşümü sonrasında, B adımında elde edilen (hiper-parametre değerleri ile eğitilmiş) modeller ile 10 katlı cross-validation yapılmıştır.

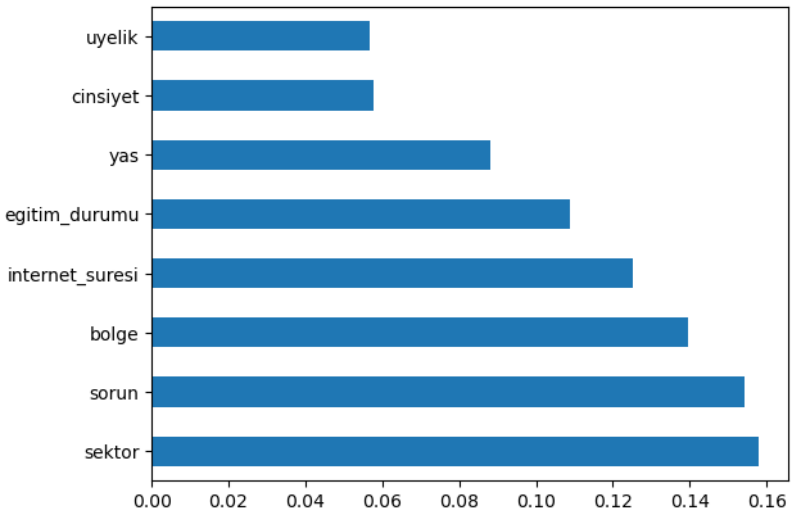


Şekil 9- PCA Dönüşümünün Performansa Etkisi

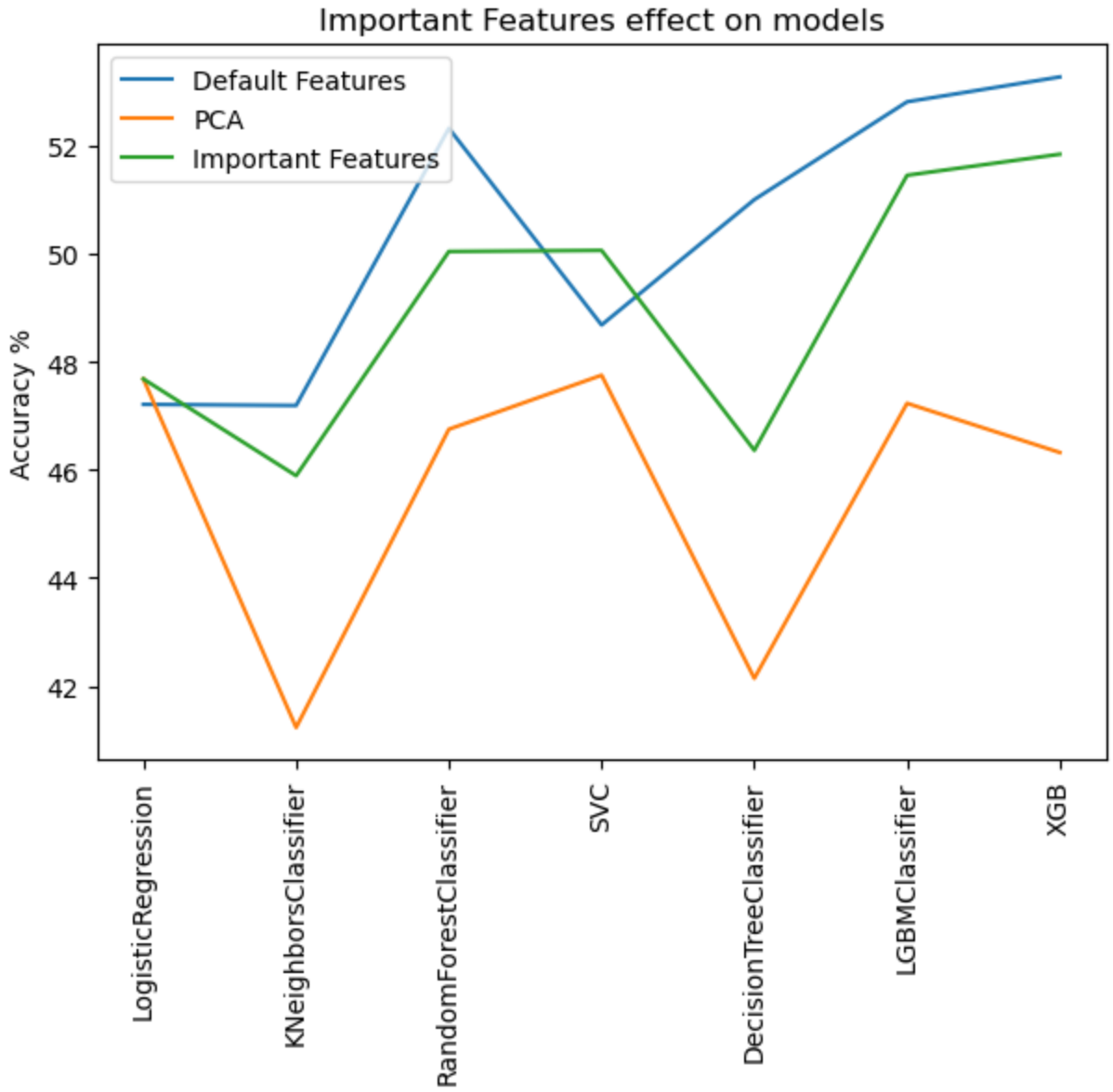
PCA dönüşümü modellerin performansını düşürmüştür. Bunun nedenlerinden biri olarak, özellik dönüşümü sırasında atılan kısımlar, modelin tahmin performansı için önemli bilgiler içeriyor olabilir.

D- Özellik Dönüşümü-2 (Feature Importances)

B bölümünde bulduğumuz ve diğerlerine göre iyi bir sonuç veren Random Forest modelinden, model performansına önemi en yüksek 8 özelliği alarak özellik dönüşümü yapılmıştır.



Şekil 10- Random Forest Modelinden Gelen En Önemli 8 Özellik



Şekil 11- Feature Importance ile Özellik Dönüşümünün Performansa Etkisi

Feature Importance yöntemiyle yapılan özellik dönüşümü sonrası elde edilen sonuçlar, PCA ile edilen sonuçlara göre daha iyi performans göstermiştir. Fakat kaybedilen özelliklerden dolayı, özellik dönüşümü yapılmadan önceki (default) haline göre genel olarak modeller kötü performans göstermiştir.

3- Yararlanılan Kaynaklar

- https://www.kaggle.com/datasets/meetnagadia/consumer-complaint-finance
- https://www.statology.org/t-value-vs-p-value/
- https://www.kaggle.com/datasets/kaushiksuresh147/customer-segmentation/code
- https://scikit-learn.org/stable/modules/cross_validation.html