Yapay Zeka Ödevi

17011080 Engin Deniz Çağlar - 18011607 Ahmet Yaman

Covid-19'a Yakalanma Yakınlığı Tespiti

Veri setimizi oluşturabilmek adına Google form oluşturduk. Oluşturduğumuz form aracılığı ile kişilerin cinsiyet, yaş, boy, kilo, haftalık kaç gün spor yaptığı, sigara kullanma durumu, kronik hastalığının varlığı, yakınlarının covid-19'a yakalanma durumu, maske kullanımı, olunan covid-19 aşı sayısı ve covid-19'a yakalanıp yakalanmadığı bilgisini aldık. Katılan 200 kişi için anlamlı veriler oluşturabilmek adına bazı veri temizlemeleri gerçekleştirdik. Kilo ve boy yazımlarında 1.82 gibi ibareleri kaldırdık.

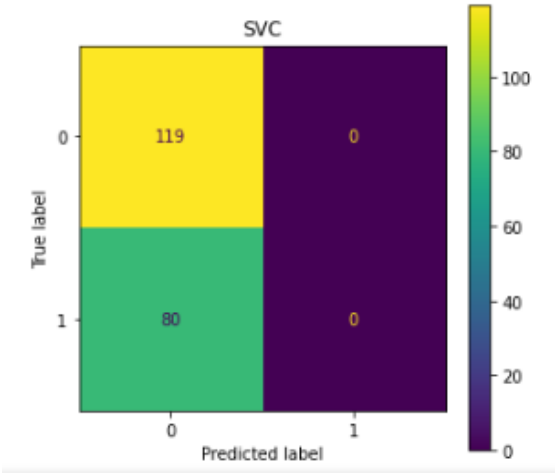
Veri temizliğinden sonra makine öğrenmesi algoritmalarında kullanabilmek için kategorileri 1 ve 0'lara çevirdik.

```
for column in columns:
    if column == 'Cinsiyet':
        dataset.loc[dataset[column] == 'Kadin', column] = 0
        dataset.loc[dataset[column] == 'Erkek', column] = 1

    if column == 'Sigara' or column == 'Kronik' or column == 'Yakinlar' or column == 'Covid19':
        dataset.loc[dataset[column] == 'Hayir', column] = 0
        dataset.loc[dataset[column] == 'Evet', column] = 1

X = dataset.iloc[:,0:-1].values
y = dataset.iloc[:,-1:].values.astype('float').ravel()
dataset.head()
```

Verileri çevirdikten sonra PCA kullanmadan 5 faklı algoritmaya soktuk.

```
def getPerformanceMetrics(X, y):
    names = ["SVC", "KNN", "RandomForest", "DTree", "Gaussian"]

    classifiers = [
        SVC(C = 5, degree = 3),
        KNeighborsClassifier(n_neighbors = 10),
        RandomForestClassifier(n_estimators=100, criterion='entropy'),
        DecisionTreeClassifier(criterion='entropy', splitter='best', max_depth=15),
        GaussianProcessClassifier(max_iter_predict = 100)
    ]


    for name, classifier in zip(names, classifiers):

        y_pred = cross_val_predict(classifier, X, y, cv=10)
        cm = confusion_matrix(y, y_pred)

        print("****************************")
        print(name," Sonuclari:")
        disp_cm = ConfusionMatrixDisplay(confusion_matrix=cm)
        fig, ax = plt.subplots(figsize=(5,5))
        disp_cm.plot(ax=ax)
        disp_cm.ax_.set_title(name)


        print("Accuracy: ",accuracy_score(y, y_pred))
        print("Recall: ",recall_score(y, y_pred, average = 'macro'))
        print("Precision: ",precision_score(y, y_pred, average ='macro'))
        print("F1 Score : ",f1_score(y, y_pred, average='macro'))
        print("Cohens Kappa : ",cohen_kappa_score(y, y_pred))
```

Algortimalar ve sonuçları:



SVC Sonuclari:

Accuracy: 0.5979899497487438

Recall: 0.5

Precision: 0.2989949748743719

F1 Score : 0.37421383647798745

Cohens Kappa : 0.0

KNN Sonuclari:

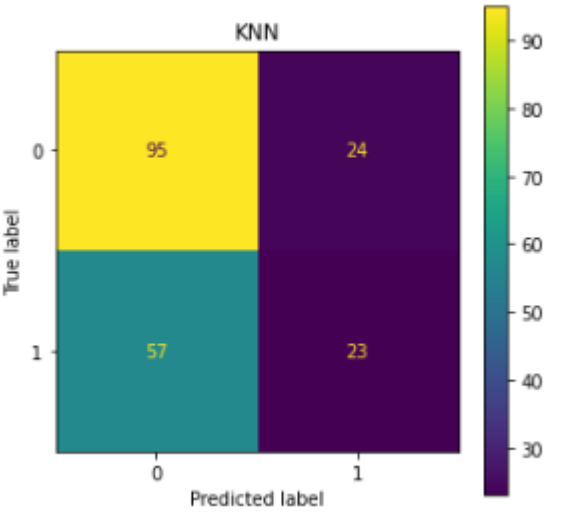
Accuracy: 0.592964824120603

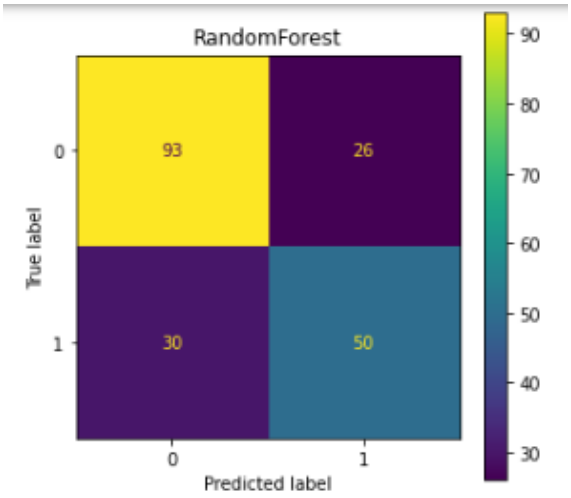
Recall: 0.5429096638655462

Precision: 0.5571808510638298

F1 Score : 0.5316558677397797

Cohens Kappa : 0.09204078183968911





RandomForest Sonuclari:

Accuracy: 0.7185929648241206

Recall: 0.7032563025210083

Precision: 0.7069961489088575

F1 Score : 0.7048103411739775

Cohens Kappa : 0.409870790086846

DTree Sonuclari:

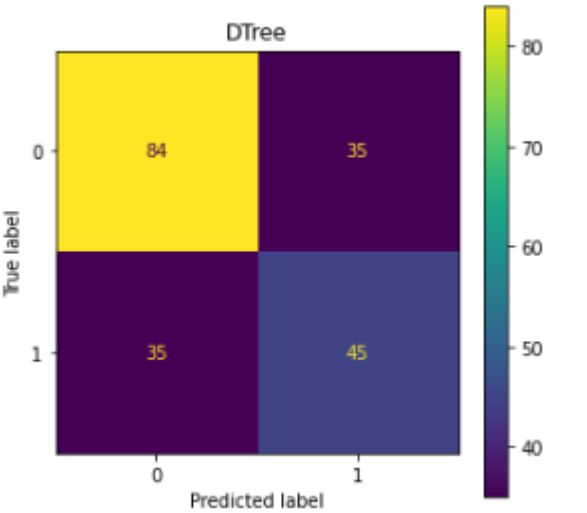
Accuracy: 0.6482412060301508

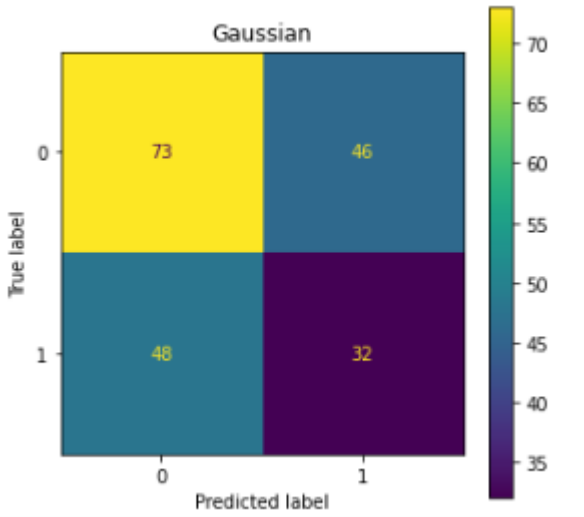
Recall: 0.6341911764705883

Precision: 0.6341911764705883

F1 Score : 0.6341911764705883

Cohens Kappa : 0.2683823529411765





Gaussian Sonuclari:

Accuracy: 0.5276381909547738

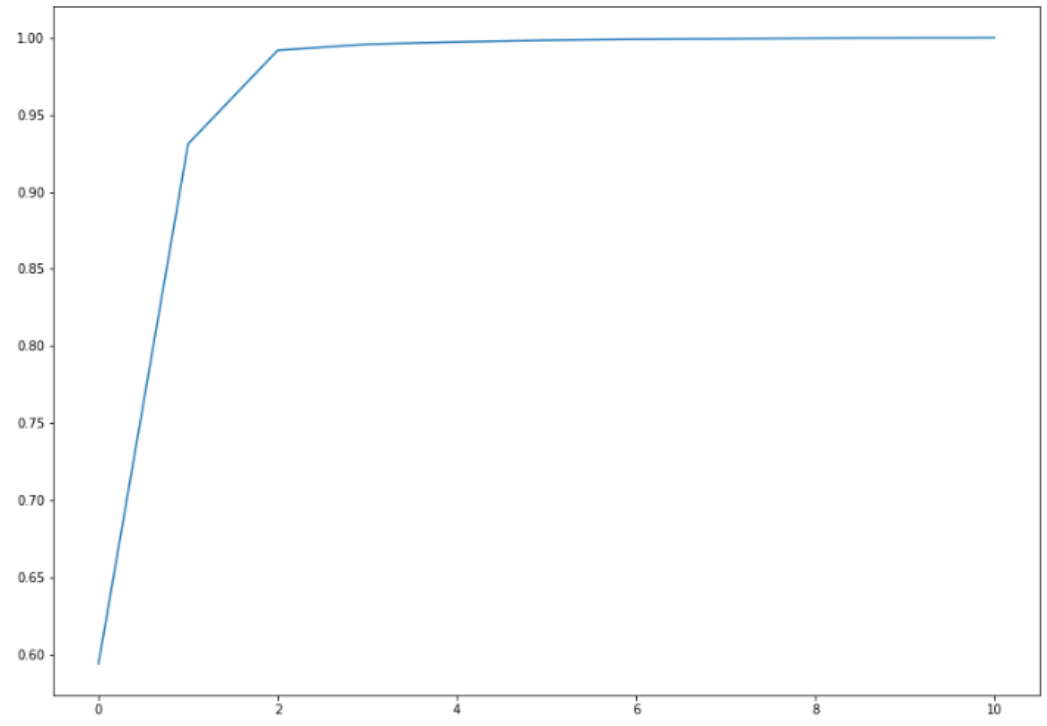
Recall: 0.5067226890756302

Precision: 0.5067810976901885

F1 Score : 0.5066983122362869

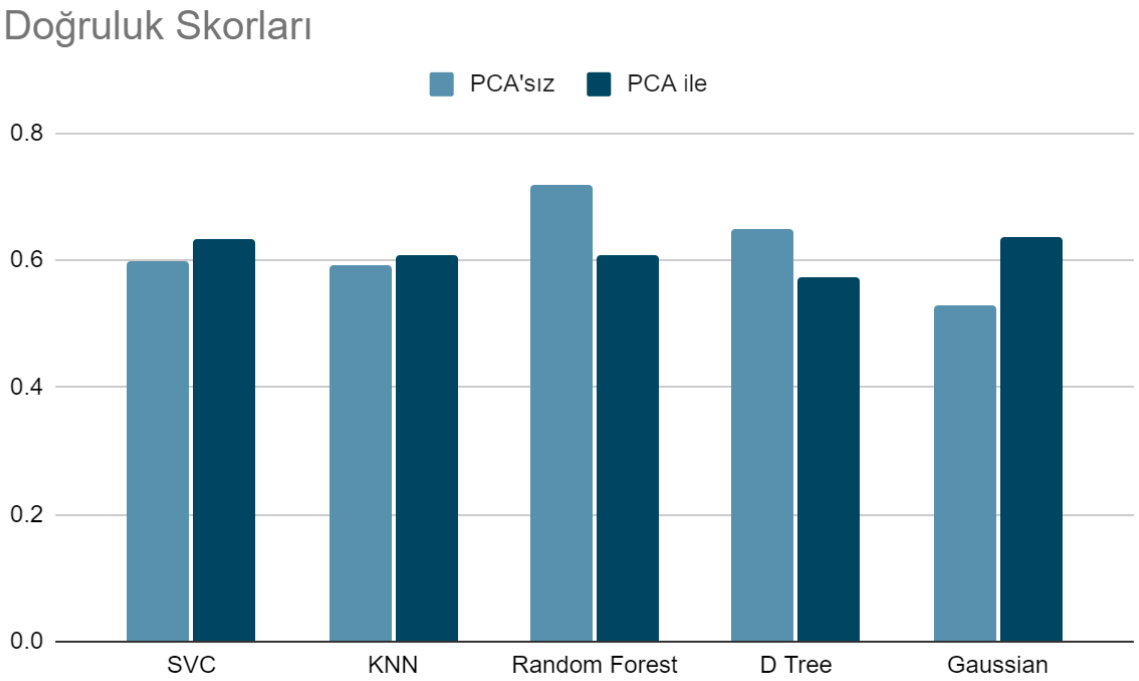
Cohens Kappa : 0.013500685581689709

PCA uyguladigimizda degiskenlerin sonuca kumulatif etkisini gosteren grafik:



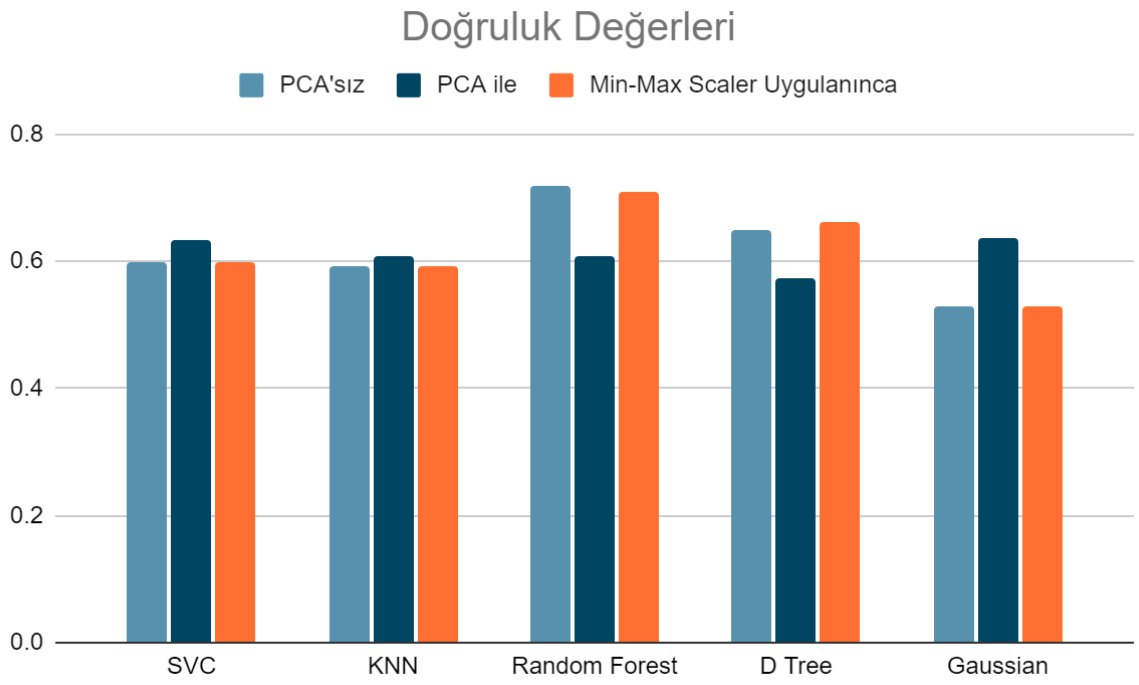
0.99 oraniyla veri setini ifade edecek sekilde PCA uyguladigimizda SVC, KNN ve Gaussian için doğruluk sonuçlarını artırdık.

```python
def pcaFitTransform(X, n_components):
    pca = sklearn.decomposition.PCA(n_components=n_components, whiten=True)
    pca.fit(X)
    X_pca = pca.transform(X)
    print("components: ",len(pca.explained_variance_ratio_))
    print("variance ratio: ", pca.explained_variance_ratio_)
    print("sum: ", sum(pca.explained_variance_ratio_))
    return X_pca

X_pca = pcaFitTransform(X, 0.95)
getPerformanceMetrics(X_pca, y)
```



MinMaxScaler ile butun verileri 0-1 araligina getirdigimizde olusan sonuclar için:

```
scaler = sklearn.preprocessing.MinMaxScaler(feature_range=(0,1))
scaled_df = pd.DataFrame(scaler.fit_transform(dataset.values),
columns=dataset.columns)

X_scaled = dataset.iloc[:,0:-1].values
y_scaled = dataset.iloc[:,-1:].values.astype('int').ravel()
getPerformanceMetrics(X_scaled, y)
```



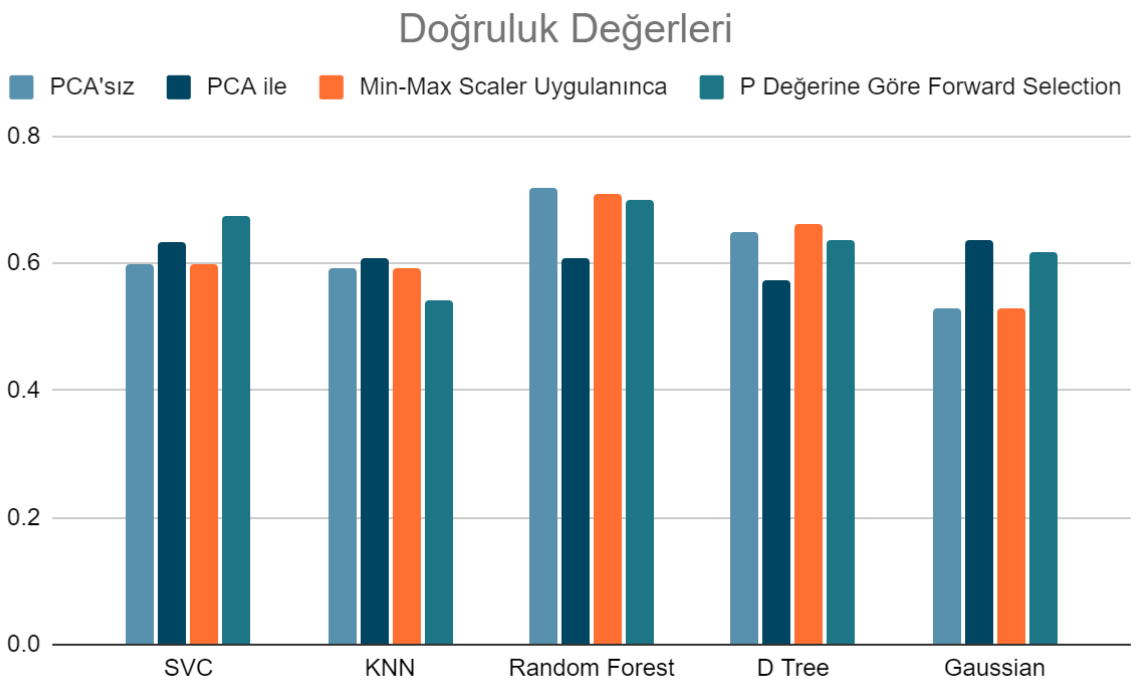
Forward Selection

```python
x1 = dataset.iloc[:,0:-1].astype(float) #features
y1 = dataset.iloc[:,-1:].values.reshape((199,)) #genres
def modelResult(x, y):

    #sabitin eklenmesi
    x = sm.add_constant(x)

    #modelin oluşturulması
    model = sm.OLS(y.astype(float), x.astype(float)).fit()

    #model performans sonuçları
    print(model.summary())
modelResult(x1,y1)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.325
Model:                            OLS   Adj. R-squared:                  0.285
Method:                 Least Squares   F-statistic:                     8.191
Date:                Wed, 25 May 2022   Prob (F-statistic):           1.22e-11
Time:                        19:38:50   Log-Likelihood:                -101.41
No. Observations:                 199   AIC:                             226.8
Df Residuals:                     187   BIC:                             266.3
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.2182      0.824     -0.265      0.791      -1.844       1.408
Cinsiyet      -0.0575      0.089     -0.643      0.521      -0.234       0.119
Yas            0.0059      0.003      2.237      0.026       0.001       0.011
Boy            0.0040      0.005      0.787      0.432      -0.006       0.014
Kilo           0.0005      0.002      0.215      0.830      -0.004       0.005
Spor          -0.0033      0.020     -0.164      0.870      -0.043       0.036
Sigara        -0.1746      0.067     -2.609      0.010      -0.307      -0.043
Kronik         0.0318      0.078      0.407      0.684      -0.122       0.186
Yakinlar       0.4429      0.061      7.251      0.000       0.322       0.563
Beslenme       0.0146      0.033      0.447      0.655      -0.050       0.079
Maske         -0.0359      0.035     -1.018      0.310      -0.105       0.034
Asi           -0.1282      0.045     -2.850      0.005      -0.217      -0.039
==============================================================================
Omnibus:                        7.855   Durbin-Watson:                   1.813
Prob(Omnibus):                  0.020   Jarque-Bera (JB):                4.163
Skew:                           0.118   Prob(JB):                        0.125
Kurtosis:                       2.332   Cond. No.                     5.26e+03
==============================================================================
```

P degerine gore forward selection ile secilen ozelliklerin sonuclari:

```
getPerformanceMetrics(X_selected, y)
```



Sonuç:

Kullandığımız 5 farklı algoritma çıktılarına göre en yüksek başarıyı Random Forest algoritmasında elde ettik. Veri setine herhangi bir müdahala etmeden elde ettiğimiz sonuçlarda %71.85 başarıya ulaştık.