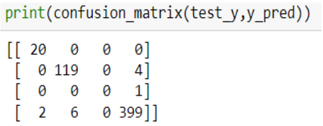
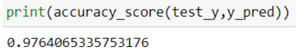
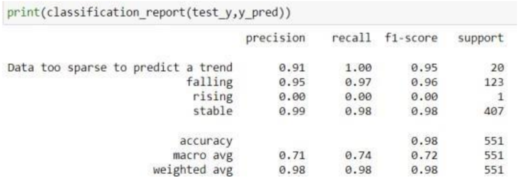
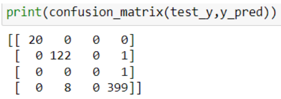
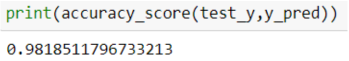
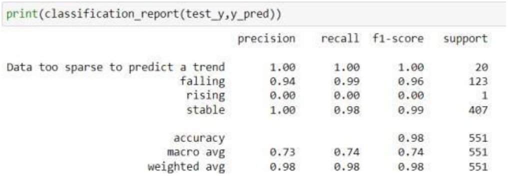
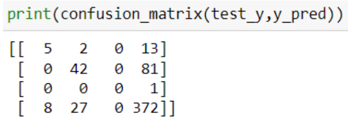
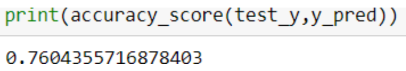
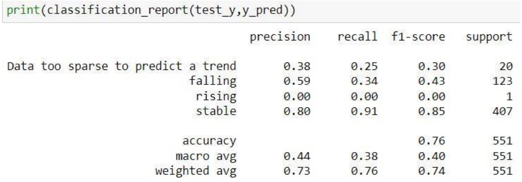
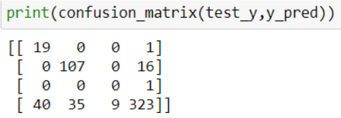
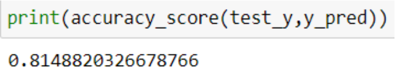
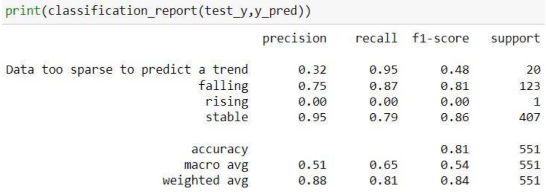
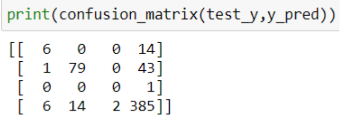
PERFORMANCE TESTING

Date	21 May 2023
Team ID	NM2023TMID17607
Project Name	Cancer Mortality and Incidence rates classification using ML

Cancer Death Rates

CANCER DEATH RATE	METRICS		
Model	CONFUSION MATRIX	ACCURACY SCORE	CLASSIFICATION REPORT
Decision Tree	 print(confusion_matrix(test_y,y_pred)) [[20 0 0 0] [0 119 0 4] [0 0 0 1] [2 6 0 399]]	 print(accuracy_score(test_y,y_pred)) 0.9764065335753176	 print(classification_report(test_y,y_pred)) precision recall f1-score support Data too sparse to predict a trend 0.91 1.00 0.95 20 falling 0.95 0.97 0.96 123 rising 0.00 0.00 0.00 1 stable 0.99 0.98 0.98 407 accuracy 0.98 551 macro avg 0.71 0.74 0.72 551 weighted avg 0.98 0.98 0.98 551
Random Forest	 print(confusion_matrix(test_y,y_pred)) [[20 0 0 0] [0 122 0 1] [0 0 0 1] [0 8 0 399]]	 print(accuracy_score(test_y,y_pred)) 0.9818511796733213	 print(classification_report(test_y,y_pred)) precision recall f1-score support Data too sparse to predict a trend 1.00 1.00 1.00 20 falling 0.94 0.99 0.96 123 rising 0.00 0.00 0.00 1 stable 1.00 0.98 0.99 407 accuracy 0.98 551 macro avg 0.73 0.74 0.74 551 weighted avg 0.98 0.98 0.98 551
K NEAREST NEIGHBORS	 print(confusion_matrix(test_y,y_pred)) [[5 2 0 13] [0 42 0 81] [0 0 0 1] [8 27 0 372]]	 print(accuracy_score(test_y,y_pred)) 0.7604355716878403	 print(classification_report(test_y,y_pred)) precision recall f1-score support Data too sparse to predict a trend 0.38 0.25 0.30 20 falling 0.59 0.34 0.43 123 rising 0.00 0.00 0.00 1 stable 0.80 0.91 0.85 407 accuracy 0.76 551 macro avg 0.44 0.38 0.40 551 weighted avg 0.73 0.76 0.74 551
Naïve Bayes	 print(confusion_matrix(test_y,y_pred)) [[19 0 0 1] [0 107 0 16] [0 0 0 1] [40 35 9 323]]	 print(accuracy_score(test_y,y_pred)) 0.8148820326678766	 print(classification_report(test_y,y_pred)) precision recall f1-score support Data too sparse to predict a trend 0.32 0.95 0.48 20 falling 0.75 0.87 0.81 123 rising 0.00 0.00 0.00 1 stable 0.95 0.79 0.86 407 accuracy 0.81 551 macro avg 0.51 0.65 0.54 551 weighted avg 0.88 0.81 0.84 551

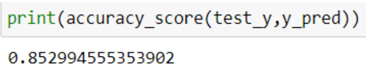
<table>
<tr>
<td>Support Vector Classification</td>
<td>



```
print(confusion_matrix(test_y,y_pred))
```

```
[[  6   0   0  14]
 [  1  79   0  43]
 [  0   0   0   1]
 [  6  14   2 385]]
```

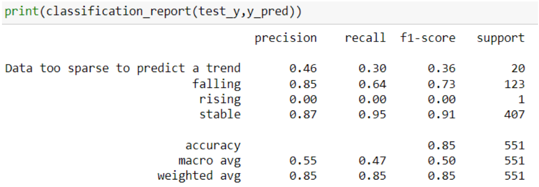
</td>
<td>



```
print(accuracy_score(test_y,y_pred))
```

```
0.852994555353902
```

</td>
<td>



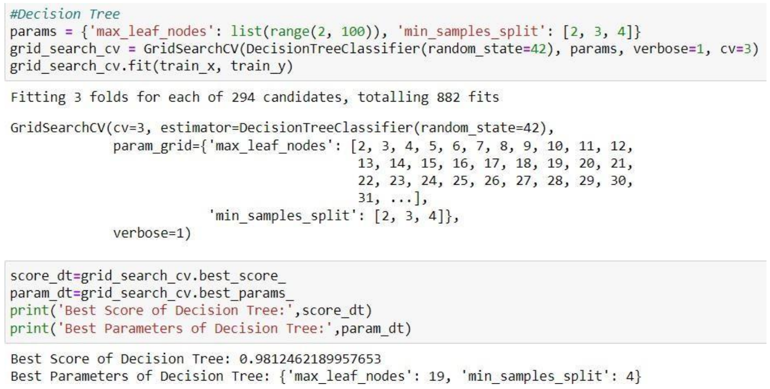
```
print(classification_report(test_y,y_pred))
```

```
                                precision    recall  f1-score   support

Data too sparse to predict a trend       0.46      0.30      0.36        20
                           falling       0.85      0.64      0.73       123
                            rising       0.00      0.00      0.00         1
                            stable       0.87      0.95      0.91       407

                          accuracy                           0.85       551
                         macro avg       0.55      0.47      0.50       551
                      weighted avg       0.85      0.85      0.85       551
```

</td>
</tr>
</table>

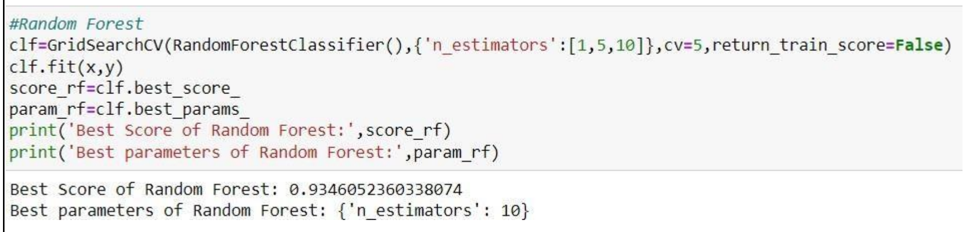
CANCER DEATH RATE	HYPERPARAMETER TUNING
MODELS	GridSearchCV

Decision Tree



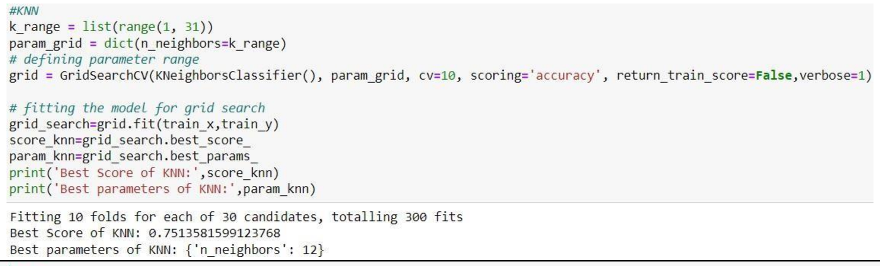
```
#Decision Tree
params = {'max_leaf_nodes': list(range(2, 100)), 'min_samples_split': [2, 3, 4]}
grid_search_cv = GridSearchCV(DecisionTreeClassifier(random_state=42), params, verbose=1, cv=3)
grid_search_cv.fit(train_x, train_y)

Fitting 3 folds for each of 294 candidates, totalling 882 fits

GridSearchCV(cv=3, estimator=DecisionTreeClassifier(random_state=42),
             param_grid={'max_leaf_nodes': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                            13, 14, 15, 16, 17, 18, 19, 20, 21,
                                            22, 23, 24, 25, 26, 27, 28, 29, 30,
                                            31, ...],
                         'min_samples_split': [2, 3, 4]},
             verbose=1)

score_dt=grid_search_cv.best_score_
param_dt=grid_search_cv.best_params_
print('Best Score of Decision Tree:',score_dt)
print('Best Parameters of Decision Tree:',param_dt)

Best Score of Decision Tree: 0.9812462189957653
Best Parameters of Decision Tree: {'max_leaf_nodes': 19, 'min_samples_split': 4}
```

Random Forest



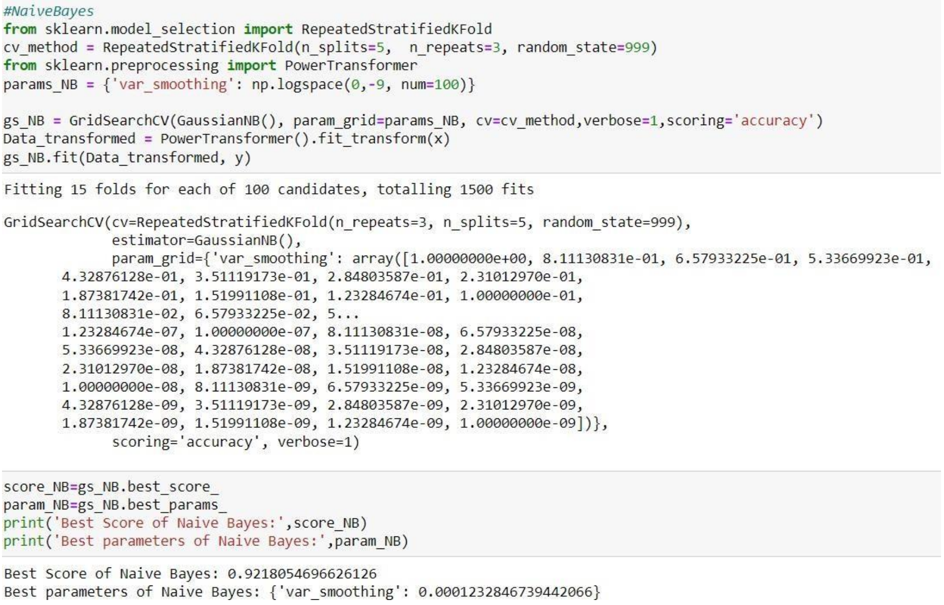
```
#Random Forest
clf=GridSearchCV(RandomForestClassifier(),{'n_estimators':[1,5,10]},cv=5,return_train_score=False)
clf.fit(x,y)
score_rf=clf.best_score_
param_rf=clf.best_params_
print('Best Score of Random Forest:',score_rf)
print('Best parameters of Random Forest:',param_rf)

Best Score of Random Forest: 0.9346052360338074
Best parameters of Random Forest: {'n_estimators': 10}
```

K Nearest Neighbors

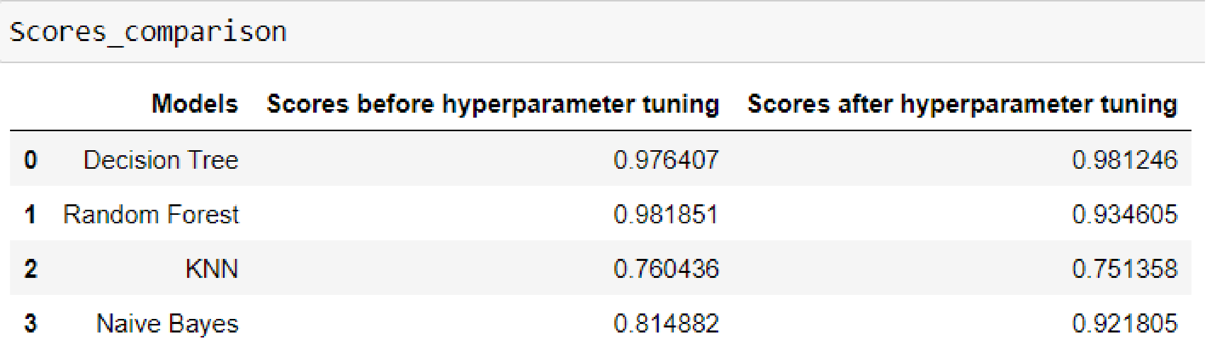


```
#KNN
k_range = list(range(1, 31))
param_grid = dict(n_neighbors=k_range)
# defining parameter range
grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=10, scoring='accuracy', return_train_score=False,verbose=1)

# fitting the model for grid search
grid_search=grid.fit(train_x,train_y)
score_knn=grid_search.best_score_
param_knn=grid_search.best_params_
print('Best Score of KNN:',score_knn)
print('Best parameters of KNN:',param_knn)

Fitting 10 folds for each of 30 candidates, totalling 300 fits
Best Score of KNN: 0.7513581599123768
Best parameters of KNN: {'n_neighbors': 12}
```

Naïve Bayes

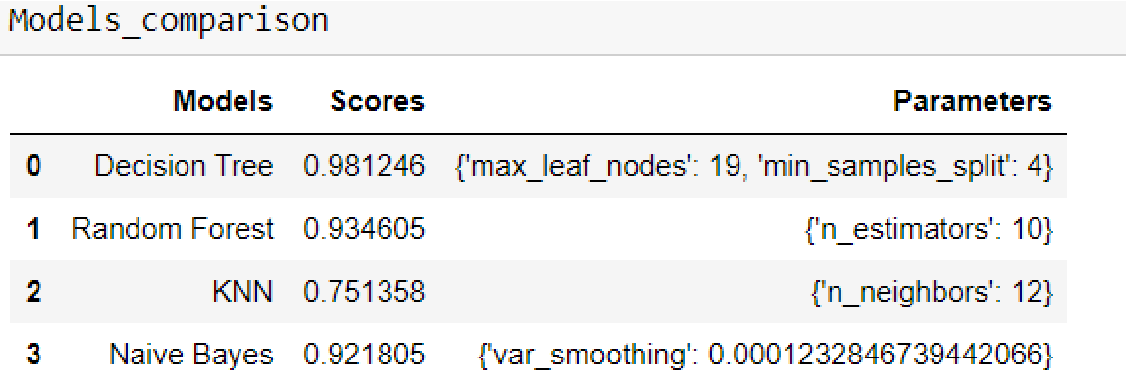


```
#NaiveBayes
from sklearn.model_selection import RepeatedStratifiedKFold
cv_method = RepeatedStratifiedKFold(n_splits=5,  n_repeats=3, random_state=999)
from sklearn.preprocessing import PowerTransformer
params_NB = {'var_smoothing': np.logspace(0,-9, num=100)}

gs_NB = GridSearchCV(GaussianNB(), param_grid=params_NB, cv=cv_method,verbose=1,scoring='accuracy')
Data_transformed = PowerTransformer().fit_transform(x)
gs_NB.fit(Data_transformed, y)

Fitting 15 folds for each of 100 candidates, totalling 1500 fits

GridSearchCV(cv=RepeatedStratifiedKFold(n_repeats=3, n_splits=5, random_state=999),
             estimator=GaussianNB(),
             param_grid={'var_smoothing': array([1.00000000e+00, 8.11130831e-01, 6.57933225e-01, 5.33669923e-01,
       4.32876128e-01, 3.51119173e-01, 2.84803587e-01, 2.31012970e-01,
       1.87381742e-01, 1.51991108e-01, 1.23284674e-01, 1.00000000e-01,
       8.11130831e-02, 6.57933225e-02, 5...
       1.23284674e-07, 1.00000000e-07, 8.11130831e-08, 6.57933225e-08,
       5.33669923e-08, 4.32876128e-08, 3.51119173e-08, 2.84803587e-08,
       2.31012970e-08, 1.87381742e-08, 1.51991108e-08, 1.23284674e-08,
       1.00000000e-08, 8.11130831e-09, 6.57933225e-09, 5.33669923e-09,
       4.32876128e-09, 3.51119173e-09, 2.84803587e-09, 2.31012970e-09,
       1.87381742e-09, 1.51991108e-09, 1.23284674e-09, 1.00000000e-09])},
             scoring='accuracy', verbose=1)

score_NB=gs_NB.best_score_
param_NB=gs_NB.best_params_
print('Best Score of Naive Bayes:',score_NB)
print('Best parameters of Naive Bayes:',param_NB)

Best Score of Naive Bayes: 0.9218054696626126
Best parameters of Naive Bayes: {'var_smoothing': 0.0001232846739442066}
```



```
Scores_comparison
```

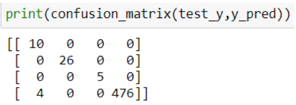
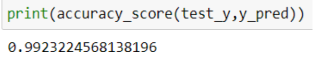
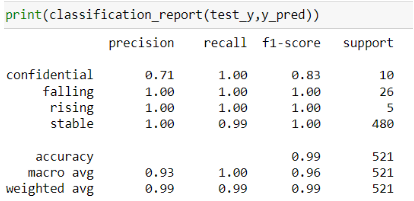
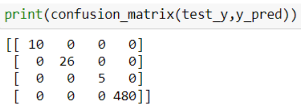
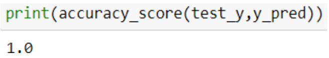
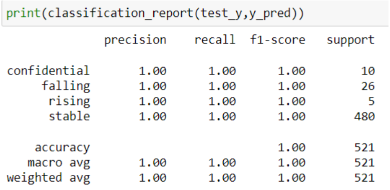
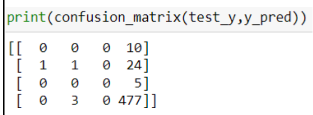
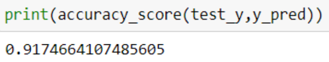
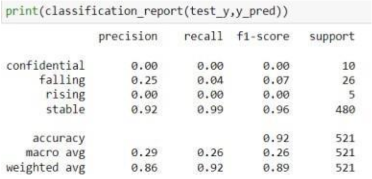
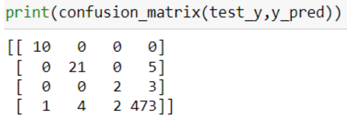
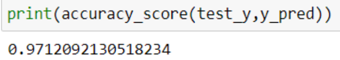
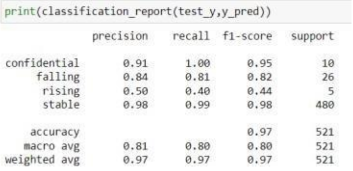
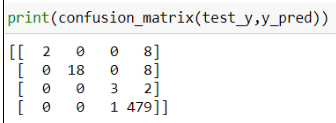
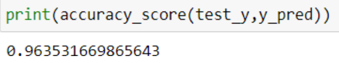
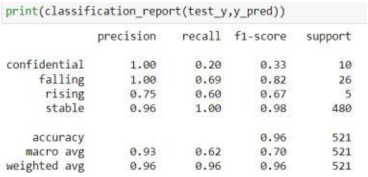
	Models	Scores before hyperparameter tuning	Scores after hyperparameter tuning
0	Decision Tree	0.976407	0.981246
1	Random Forest	0.981851	0.934605
2	KNN	0.760436	0.751358
3	Naive Bayes	0.814882	0.921805



```
Models_comparison
```

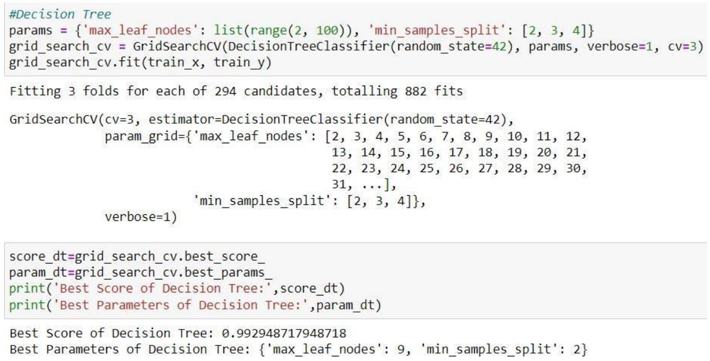
	Models	Scores	Parameters
0	Decision Tree	0.981246	{'max_leaf_nodes': 19, 'min_samples_split': 4}
1	Random Forest	0.934605	{'n_estimators': 10}
2	KNN	0.751358	{'n_neighbors': 12}
3	Naive Bayes	0.921805	{'var_smoothing': 0.0001232846739442066}

Cancer Incidence Rates

CANCER INCIDENCE RATES	METRICS		
MODELS	CONFUSION MATRIX	ACCURACY SCORE	CLASSIFICATION REPORT
Decision Tree	 print(confusion_matrix(test_y,y_pred)) [[10 0 0 0] [0 26 0 0] [0 0 5 0] [4 0 0 476]]	 print(accuracy_score(test_y,y_pred)) 0.9923224568138196	 print(classification_report(test_y,y_pred)) precision recall f1-score support confidential 0.71 1.00 0.83 10 falling 1.00 1.00 1.00 26 rising 1.00 1.00 1.00 5 stable 1.00 0.99 1.00 480 accuracy 0.99 521 macro avg 0.93 1.00 0.96 521 weighted avg 0.99 0.99 0.99 521
Random Forest	 print(confusion_matrix(test_y,y_pred)) [[10 0 0 0] [0 26 0 0] [0 0 5 0] [0 0 0 480]]	 print(accuracy_score(test_y,y_pred)) 1.0	 print(classification_report(test_y,y_pred)) precision recall f1-score support confidential 1.00 1.00 1.00 10 falling 1.00 1.00 1.00 26 rising 1.00 1.00 1.00 5 stable 1.00 1.00 1.00 480 accuracy 1.00 521 macro avg 1.00 1.00 1.00 521 weighted avg 1.00 1.00 1.00 521
K Nearest Neighbors	 print(confusion_matrix(test_y,y_pred)) [[0 0 0 10] [1 1 0 24] [0 0 0 5] [0 3 0 477]]	 print(accuracy_score(test_y,y_pred)) 0.9174664107485605	 print(classification_report(test_y,y_pred)) precision recall f1-score support confidential 0.00 0.00 0.00 10 falling 0.25 0.04 0.07 26 rising 0.00 0.00 0.00 5 stable 0.92 0.99 0.96 480 accuracy 0.92 521 macro avg 0.29 0.26 0.26 521 weighted avg 0.86 0.92 0.89 521
Naïve Bayes	 print(confusion_matrix(test_y,y_pred)) [[10 0 0 0] [0 21 0 5] [0 0 2 3] [1 4 2 473]]	 print(accuracy_score(test_y,y_pred)) 0.9712092130518234	 print(classification_report(test_y,y_pred)) precision recall f1-score support confidential 0.91 1.00 0.95 10 falling 0.84 0.81 0.82 26 rising 0.50 0.40 0.44 5 stable 0.98 0.99 0.98 480 accuracy 0.97 521 macro avg 0.81 0.80 0.80 521 weighted avg 0.97 0.97 0.97 521
Support Vector Classification	 print(confusion_matrix(test_y,y_pred)) [[2 0 0 8] [0 18 0 8] [0 0 3 2] [0 0 1 479]]	 print(accuracy_score(test_y,y_pred)) 0.963531669865643	 print(classification_report(test_y,y_pred)) precision recall f1-score support confidential 1.00 0.20 0.33 10 falling 1.00 0.69 0.82 26 rising 0.75 0.60 0.67 5 stable 0.96 1.00 0.98 480 accuracy 0.96 521 macro avg 0.93 0.62 0.70 521 weighted avg 0.96 0.96 0.96 521

CANCER INCIDENCE RATES	HYPERPARAMETER TUNING
MODELS	GridSearchCV

Decision Tree



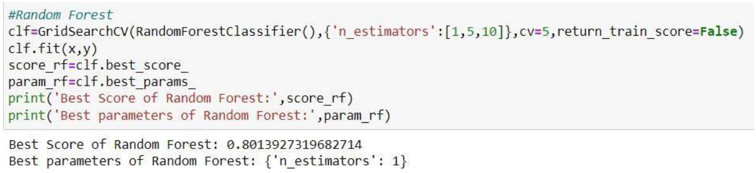
```
#Decision Tree
params = {'max_leaf_nodes': list(range(2, 100)), 'min_samples_split': [2, 3, 4]}
grid_search_cv = GridSearchCV(DecisionTreeClassifier(random_state=42), params, verbose=1, cv=3)
grid_search_cv.fit(train_x, train_y)
```

```
Fitting 3 folds for each of 294 candidates, totalling 882 fits

GridSearchCV(cv=3, estimator=DecisionTreeClassifier(random_state=42),
             param_grid={'max_leaf_nodes': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                            13, 14, 15, 16, 17, 18, 19, 20, 21,
                                            22, 23, 24, 25, 26, 27, 28, 29, 30,
                                            31, ...],
                         'min_samples_split': [2, 3, 4]},
             verbose=1)
```

```
score_dt=grid_search_cv.best_score_
param_dt=grid_search_cv.best_params_
print('Best Score of Decision Tree:',score_dt)
print('Best Parameters of Decision Tree:',param_dt)
```

```
Best Score of Decision Tree: 0.992948717948718
Best Parameters of Decision Tree: {'max_leaf_nodes': 9, 'min_samples_split': 2}
```

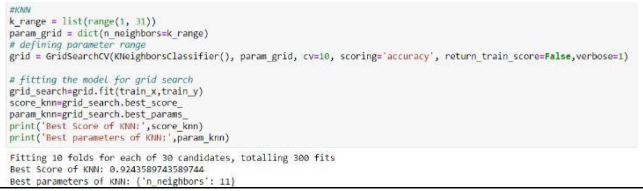
Random Forest



```
#Random Forest
clf=GridSearchCV(RandomForestClassifier(),{'n_estimators':[1,5,10]},cv=5,return_train_score=False)
clf.fit(x,y)
score_rf=clf.best_score_
param_rf=clf.best_params_
print('Best Score of Random Forest:',score_rf)
print('Best parameters of Random Forest:',param_rf)
```

```
Best Score of Random Forest: 0.8013927319682714
Best parameters of Random Forest: {'n_estimators': 1}
```

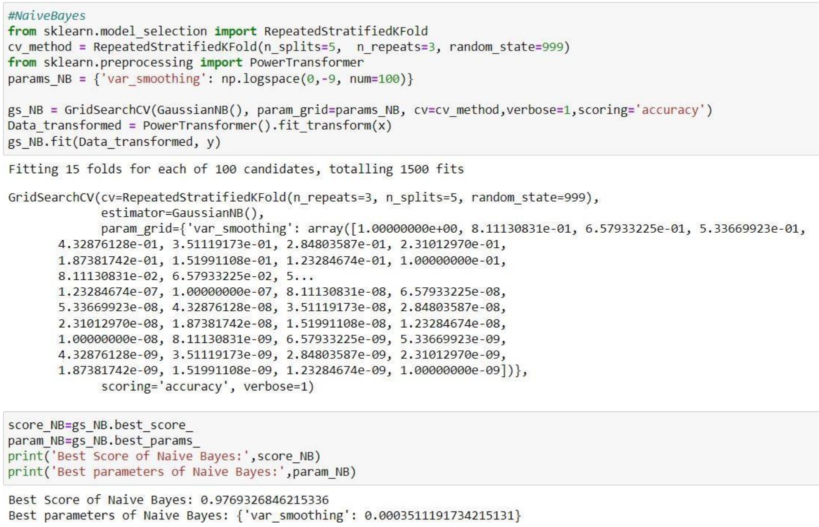
K Nearest Neighbors



```
#KNN
k_range = list(range(1, 31))
param_grid = dict(n_neighbors=k_range)
# defining parameter range
grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=10, scoring='accuracy', return_train_score=False,verbose=1)

# fitting the model for grid search
grid_search=grid.fit(train_x,train_y)
score_knn=grid_search.best_score_
param_knn=grid_search.best_params_
print('Best Score of KNN:',score_knn)
print('Best parameters of KNN:',param_knn)
```

```
Fitting 10 folds for each of 30 candidates, totalling 300 fits
Best Score of KNN: 0.9243589743589744
Best parameters of KNN: {'n_neighbors': 11}
```

Naïve Bayes



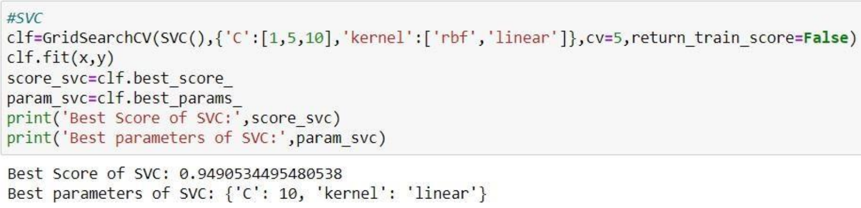
```
#NaiveBayes
from sklearn.model_selection import RepeatedStratifiedKFold
cv_method = RepeatedStratifiedKFold(n_splits=5,  n_repeats=3, random_state=999)
from sklearn.preprocessing import PowerTransformer
params_NB = {'var_smoothing': np.logspace(0,-9, num=100)}

gs_NB = GridSearchCV(GaussianNB(), param_grid=params_NB, cv=cv_method,verbose=1,scoring='accuracy')
Data_transformed = PowerTransformer().fit_transform(x)
gs_NB.fit(Data_transformed, y)
```

```
Fitting 15 folds for each of 100 candidates, totalling 1500 fits

GridSearchCV(cv=RepeatedStratifiedKFold(n_repeats=3, n_splits=5, random_state=999),
             estimator=GaussianNB(),
             param_grid={'var_smoothing': array([1.00000000e+00, 8.11130831e-01, 6.57933225e-01, 5.33669923e-01,
       4.32876128e-01, 3.51119173e-01, 2.84803587e-01, 2.31012970e-01,
       1.87381742e-01, 1.51991108e-01, 1.23284674e-01, 1.00000000e-01,
       8.11130831e-02, 6.57933225e-02, 5...
       1.23284674e-07, 1.00000000e-07, 8.11130831e-08, 6.57933225e-08,
       5.33669923e-08, 4.32876128e-08, 3.51119173e-08, 2.84803587e-08,
       2.31012970e-08, 1.87381742e-08, 1.51991108e-08, 1.23284674e-08,
       1.00000000e-08, 8.11130831e-09, 6.57933225e-09, 5.33669923e-09,
       4.32876128e-09, 3.51119173e-09, 2.84803587e-09, 2.31012970e-09,
       1.87381742e-09, 1.51991108e-09, 1.23284674e-09, 1.00000000e-09])},
             scoring='accuracy', verbose=1)
```

```
score_NB=gs_NB.best_score_
param_NB=gs_NB.best_params_
print('Best Score of Naive Bayes:',score_NB)
print('Best parameters of Naive Bayes:',param_NB)
```

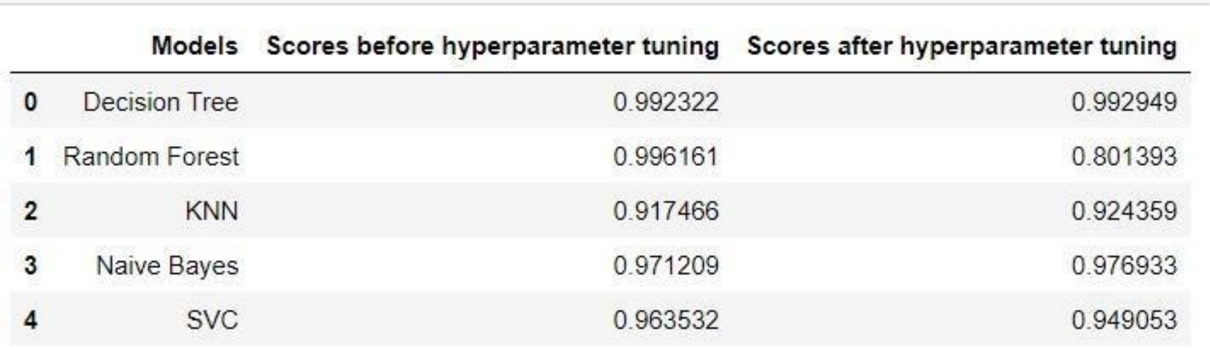
```
Best Score of Naive Bayes: 0.9769326846215336
Best parameters of Naive Bayes: {'var_smoothing': 0.0003511191734215131}
```

Support Vector Classification

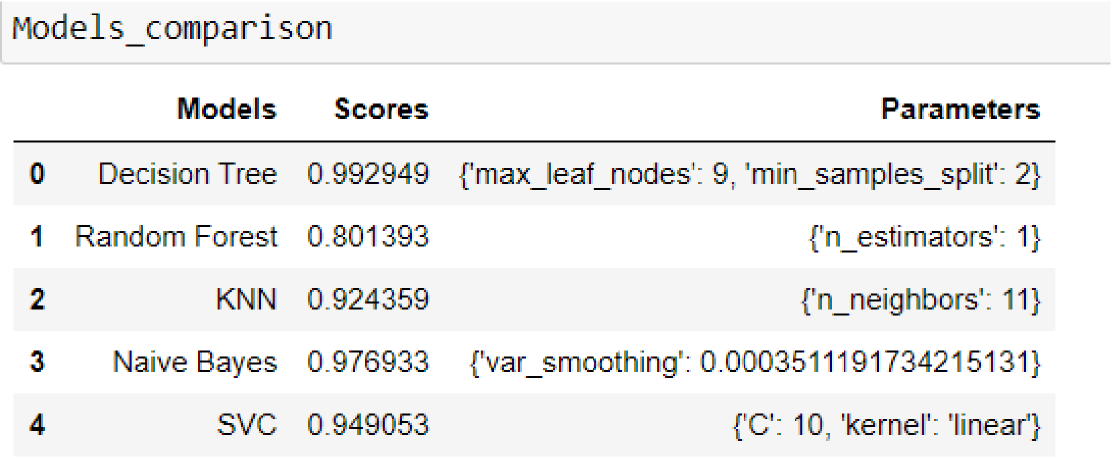


```
#SVC
clf=GridSearchCV(SVC(),{'C':[1,5,10],'kernel':['rbf','linear']},cv=5,return_train_score=False)
clf.fit(x,y)
score_svc=clf.best_score_
param_svc=clf.best_params_
print('Best Score of SVC:',score_svc)
print('Best parameters of SVC:',param_svc)
```

```
Best Score of SVC: 0.9490534495480538
Best parameters of SVC: {'C': 10, 'kernel': 'linear'}
```



	Models	Scores before hyperparameter tuning	Scores after hyperparameter tuning
0	Decision Tree	0.992322	0.992949
1	Random Forest	0.996161	0.801393
2	KNN	0.917466	0.924359
3	Naive Bayes	0.971209	0.976933
4	SVC	0.963532	0.949053



```
Models_comparison
```

	Models	Scores	Parameters
0	Decision Tree	0.992949	{'max_leaf_nodes': 9, 'min_samples_split': 2}
1	Random Forest	0.801393	{'n_estimators': 1}
2	KNN	0.924359	{'n_neighbors': 11}
3	Naive Bayes	0.976933	{'var_smoothing': 0.0003511191734215131}
4	SVC	0.949053	{'C': 10, 'kernel': 'linear'}