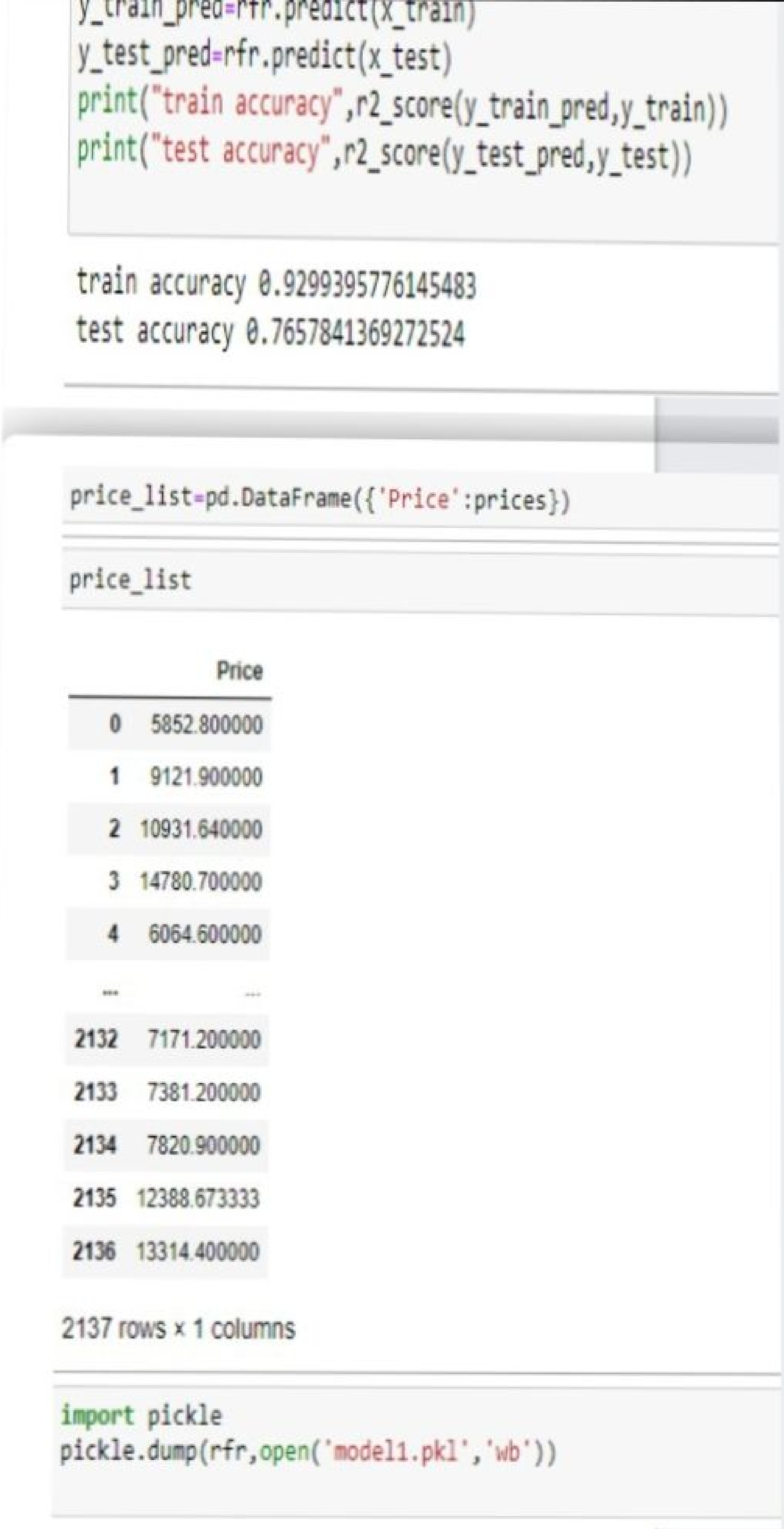


```
y_train_pred=rfr.predict(x_train)
y_test_pred=rfr.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
train accuracy 0.9299395776145483
test accuracy 0.7657841369272524
```

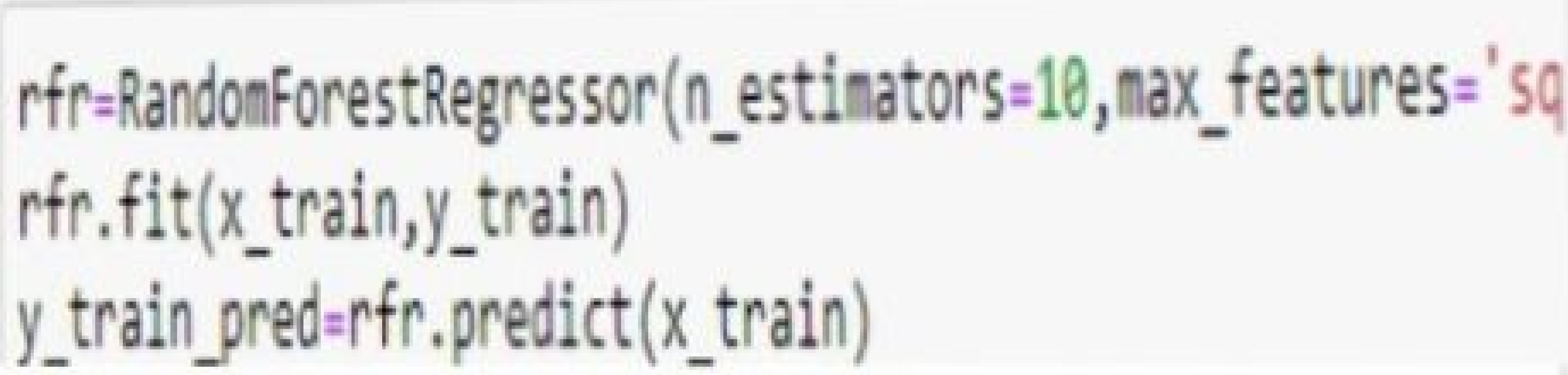
```
price_list=pd.DataFrame({'Price':prices})
```

```
price_list
```

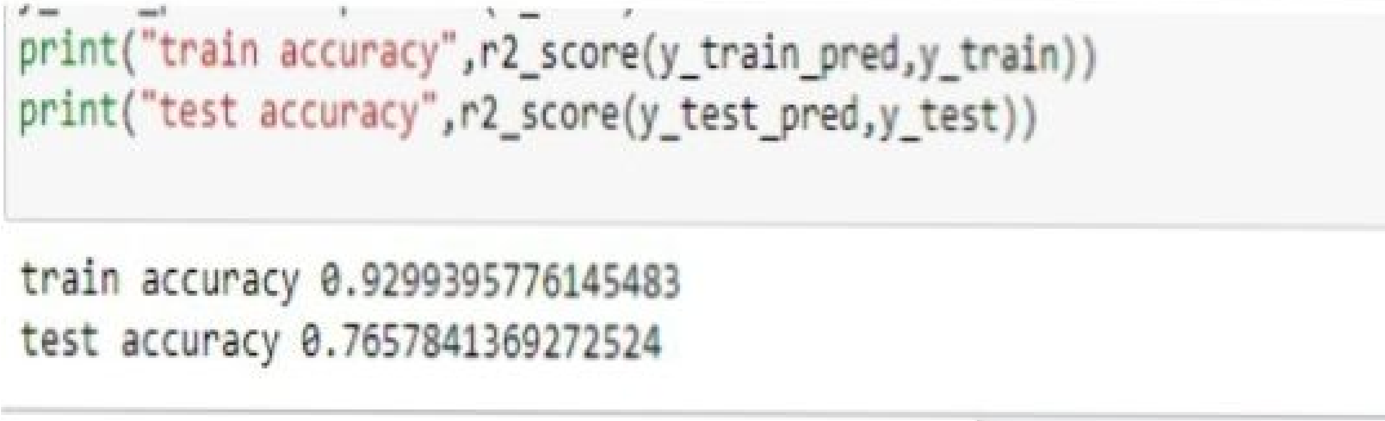
| | Price |
|---|---|
| 0 | 5852.800000 |
| 1 | 9121.900000 |
| 2 | 10931.640000 |
| 3 | 14780.700000 |
| 4 | 6064.600000 |
| ... | ... |
| 2132 | 7171.200000 |
| 2133 | 7381.200000 |
| 2134 | 7820.900000 |
| 2135 | 12388.673333 |
| 2136 | 13314.400000 |

2137 rows × 1 columns

```
import pickle
pickle.dump(rfr,open('model1.pkl','wb'))
```



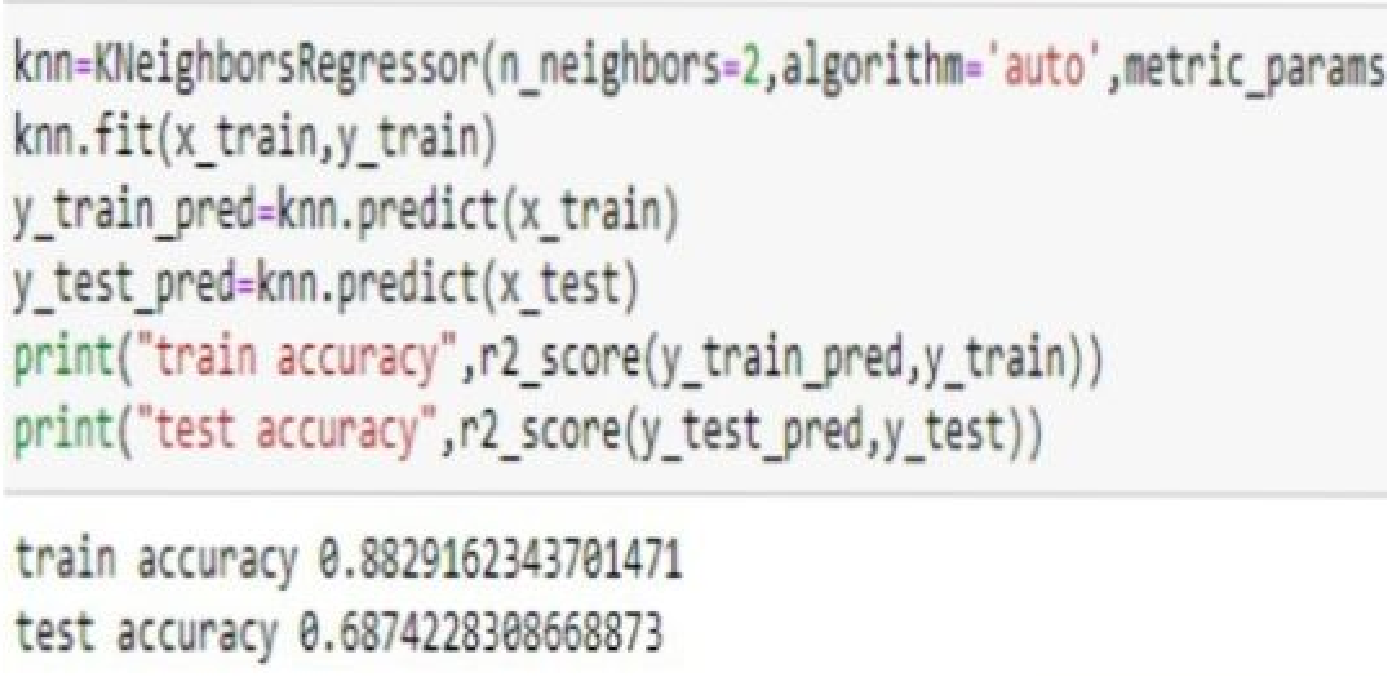
```
rfr=RandomForestRegressor(n_estimators=10,max_features='sq
rfr.fit(x_train,y_train)
y_train_pred=rfr.predict(x_train)
```



```
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

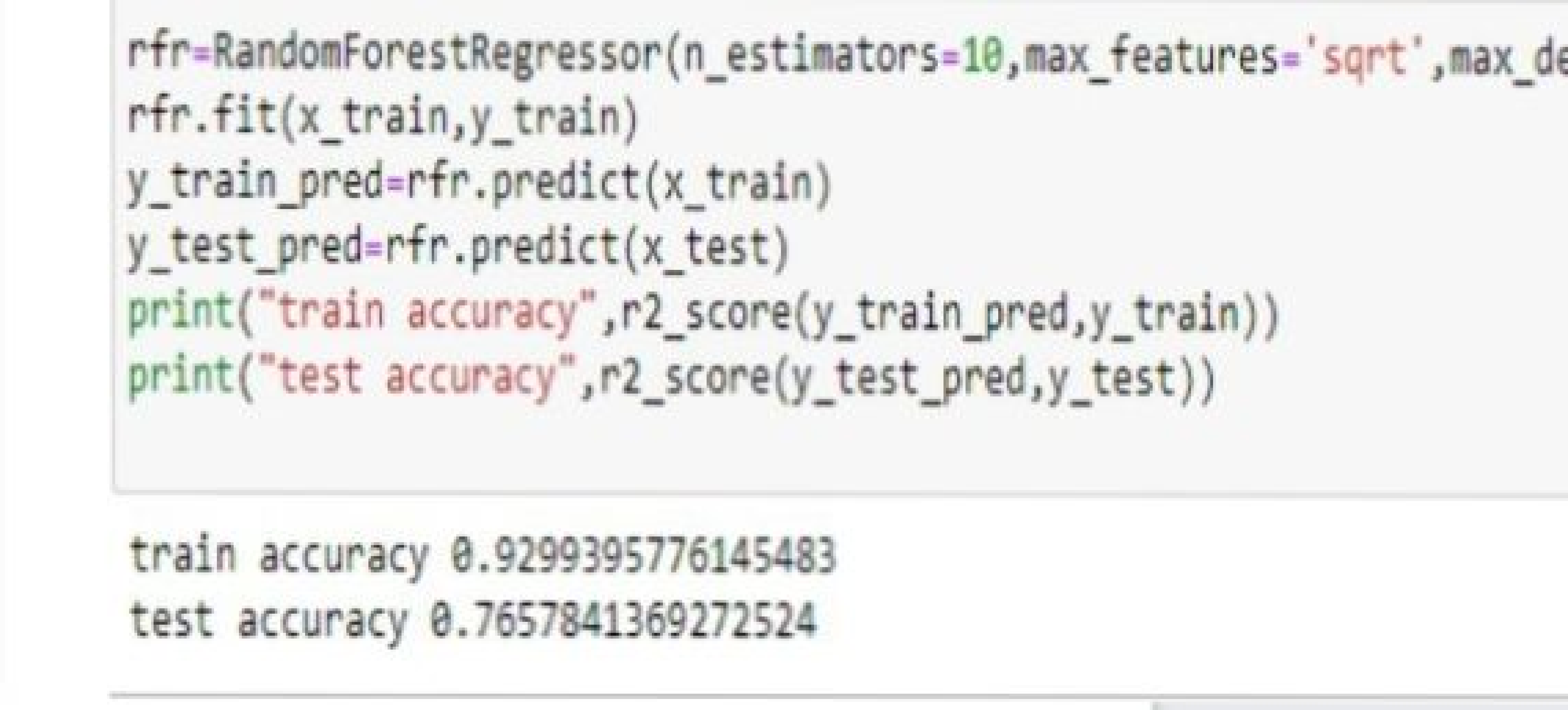
```
train accuracy 0.9299395776145483
test accuracy 0.7657841369272524
```

# Checking Train and Test Accuracy by RandomSearchCV using KNN  Model2



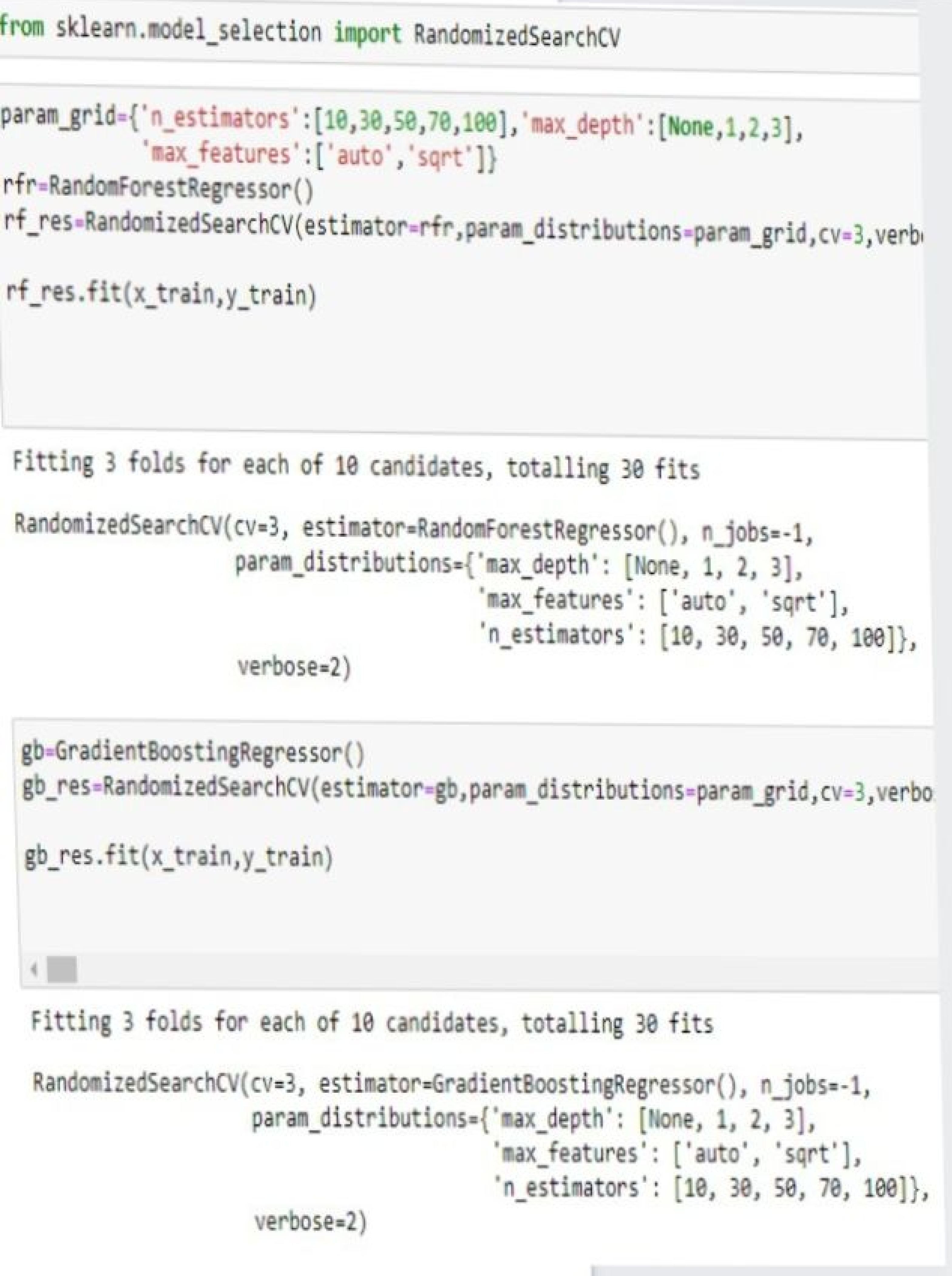
```
knn=KNeighborsRegressor(n_neighbors=2,algorithm='auto',metric_params
knn.fit(x_train,y_train)
y_train_pred=knn.predict(x_train)
y_test_pred=knn.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
train accuracy 0.8829162343701471
test accuracy 0.6874228308668873
```



```
rfr=RandomForestRegressor(n_estimators=10,max_features='sqrt',max_de
rfr.fit(x_train,y_train)
y_train_pred=rfr.predict(x_train)
y_test_pred=rfr.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
train accuracy 0.9299395776145483
test accuracy 0.7657841369272524
```



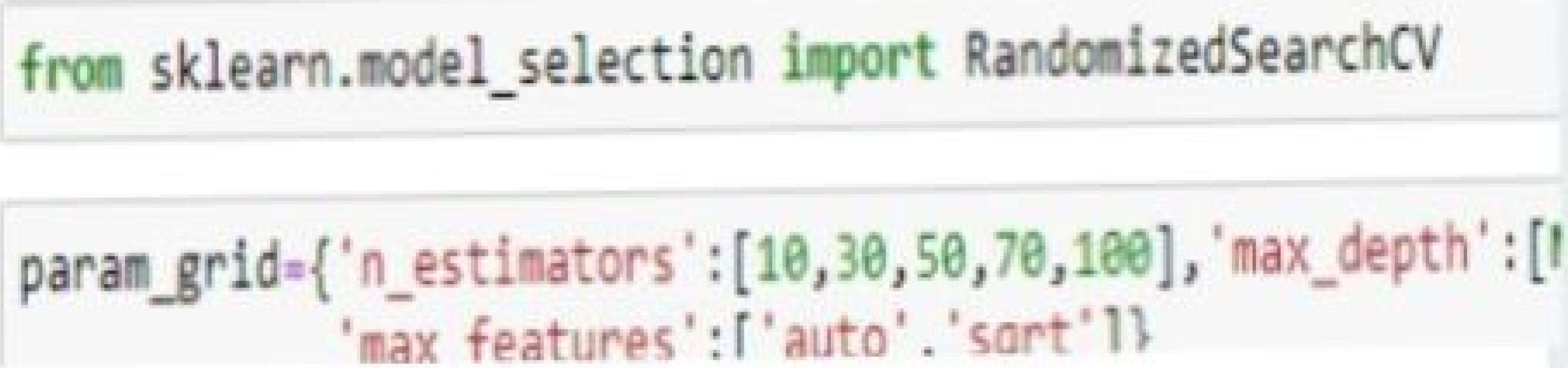
```
from sklearn.model_selection import RandomizedSearchCV
```

```
param_grid={'n_estimators':[10,30,50,70,100],'max_depth':[None,1,2,3],
            'max_features':['auto','sqrt']}
rfr=RandomForestRegressor()
rf_res=RandomizedSearchCV(estimator=rfr,param_distributions=param_grid,cv=3,verb

rf_res.fit(x_train,y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=-1,
                   param_distributions={'max_depth': [None, 1, 2, 3],
                                        'max_features': ['auto', 'sqrt'],
                                        'n_estimators': [10, 30, 50, 70, 100]},
                   verbose=2)
```

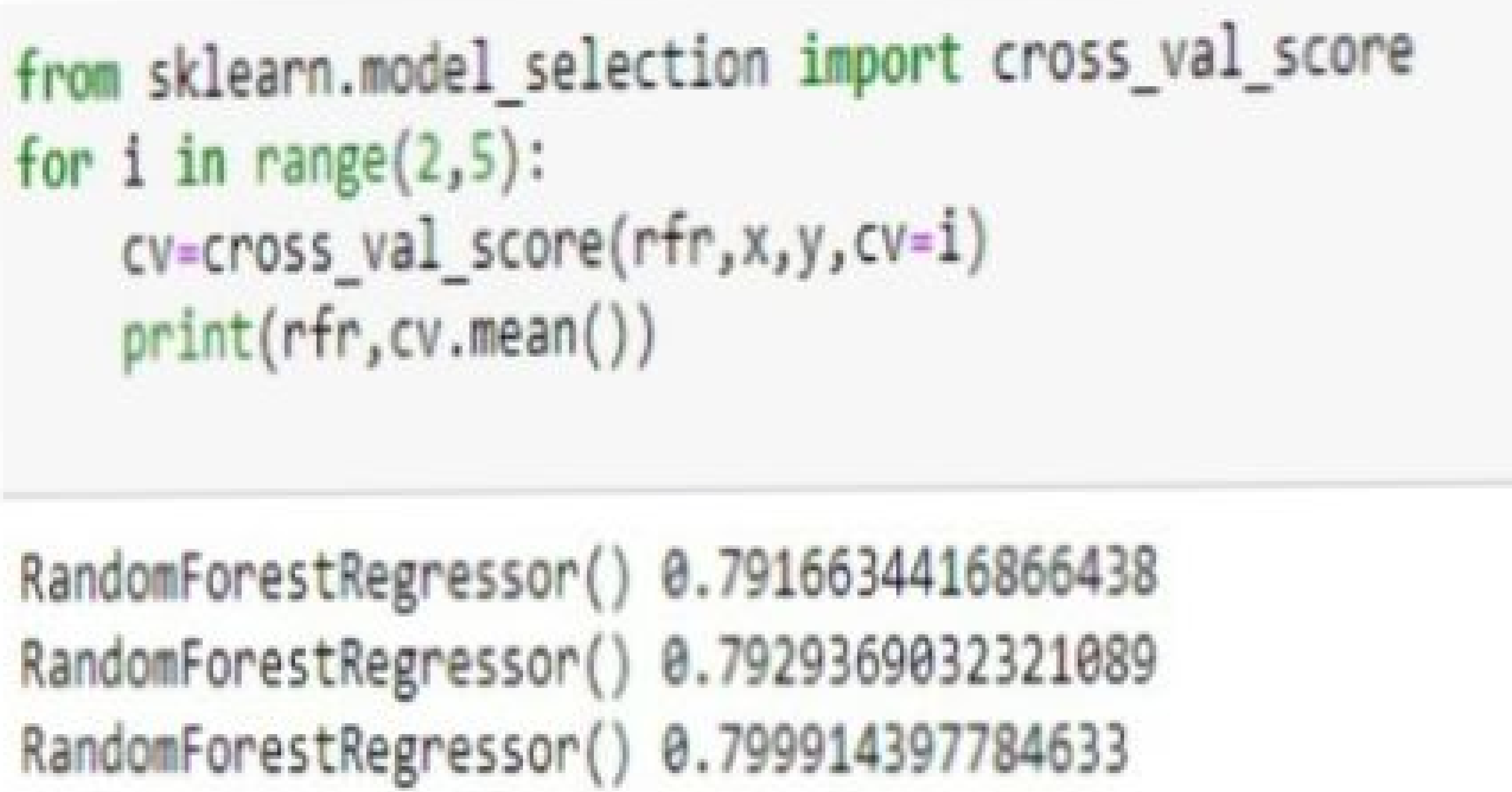
```
gb=GradientBoostingRegressor()
gb_res=RandomizedSearchCV(estimator=gb,param_distributions=param_grid,cv=3,verbo

gb_res.fit(x_train,y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

RandomizedSearchCV(cv=3, estimator=GradientBoostingRegressor(), n_jobs=-1,
                   param_distributions={'max_depth': [None, 1, 2, 3],
                                        'max_features': ['auto', 'sqrt'],
                                        'n_estimators': [10, 30, 50, 70, 100]},
                   verbose=2)
```



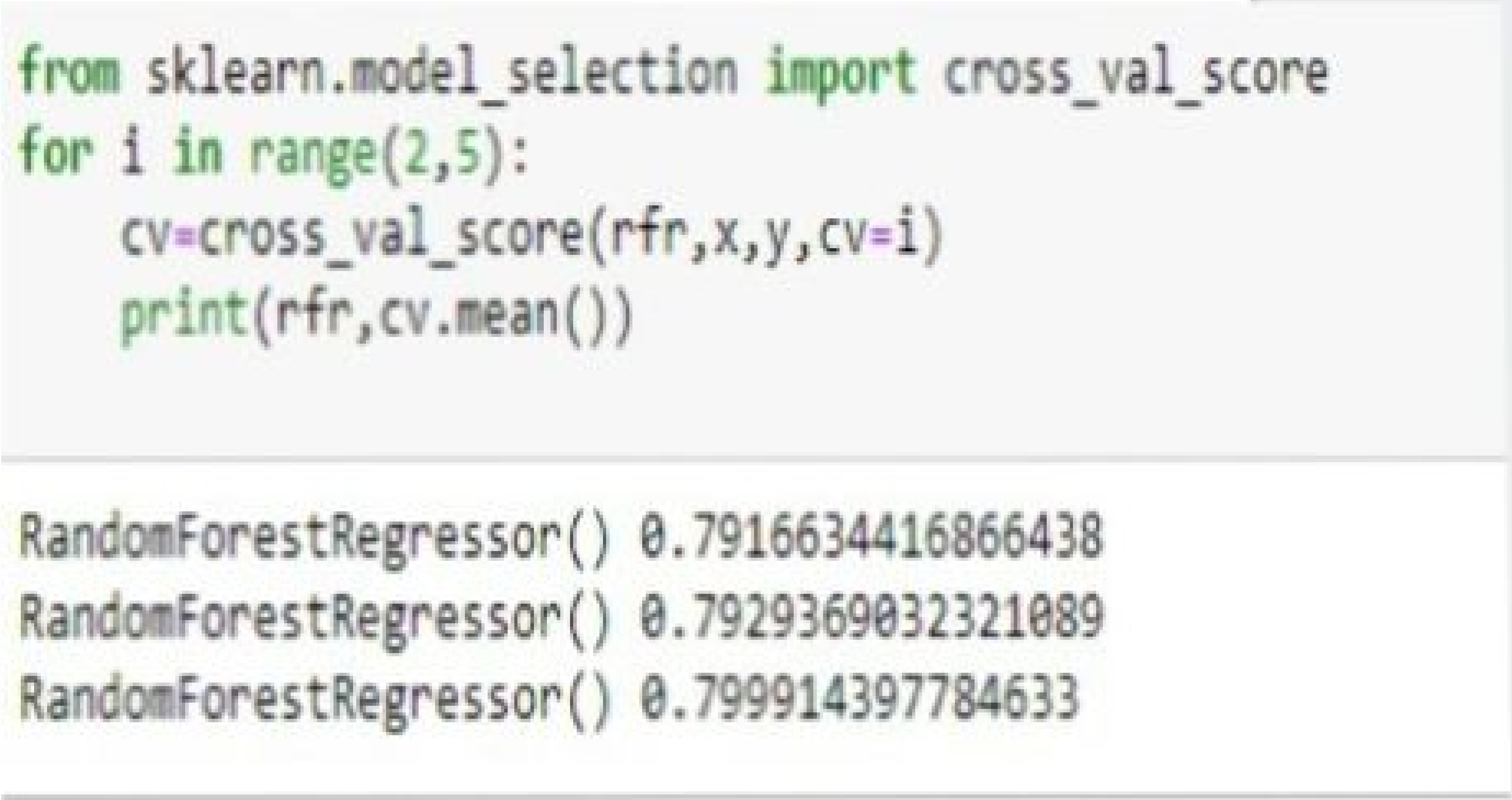
```
from sklearn.model_selection import RandomizedSearchCV
```

```
param_grid={'n_estimators':[10,30,50,70,100],'max_depth':[
            'max_features':['auto','sqrt']}
```



```
from sklearn.model_selection import cross_val_score
for i in range(2,5):
    cv=cross_val_score(rfr,x,y,cv=i)
    print(rfr,cv.mean())
```

```
RandomForestRegressor() 0.7916634416866438
RandomForestRegressor() 0.7929369032321089
RandomForestRegressor() 0.799914397784633
```



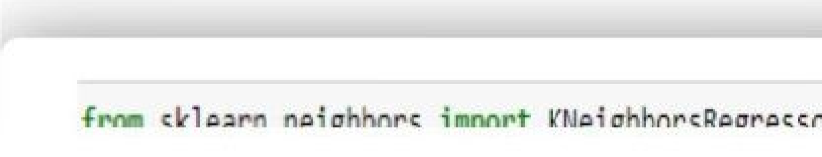
```
from sklearn.model_selection import cross_val_score
for i in range(2,5):
    cv=cross_val_score(rfr,x,y,cv=i)
    print(rfr,cv.mean())
```

```
RandomForestRegressor() 0.7916634416866438
RandomForestRegressor() 0.7929369032321089
RandomForestRegressor() 0.799914397784633
```

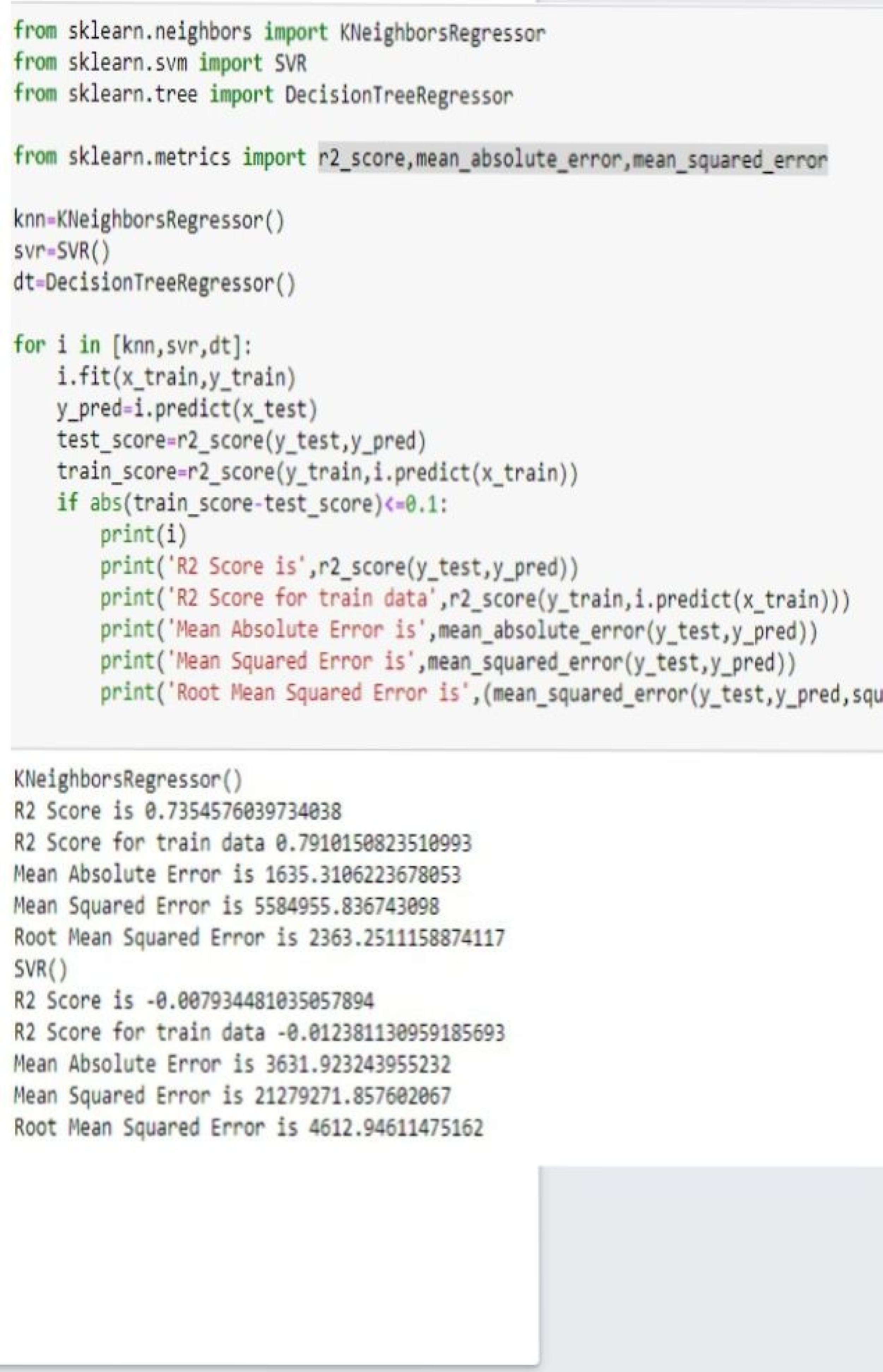
# Regression Model

### KNeighborsRegressor, SVR, DecisionTreeRegressor

A function named KNN, SVR, DecisionTree is created and train and test data are passed as the parameters. Inside the function, KNN, SVR, DecisionTree algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, r2_score, mean_absolute_error, and mean_squared_error is done.

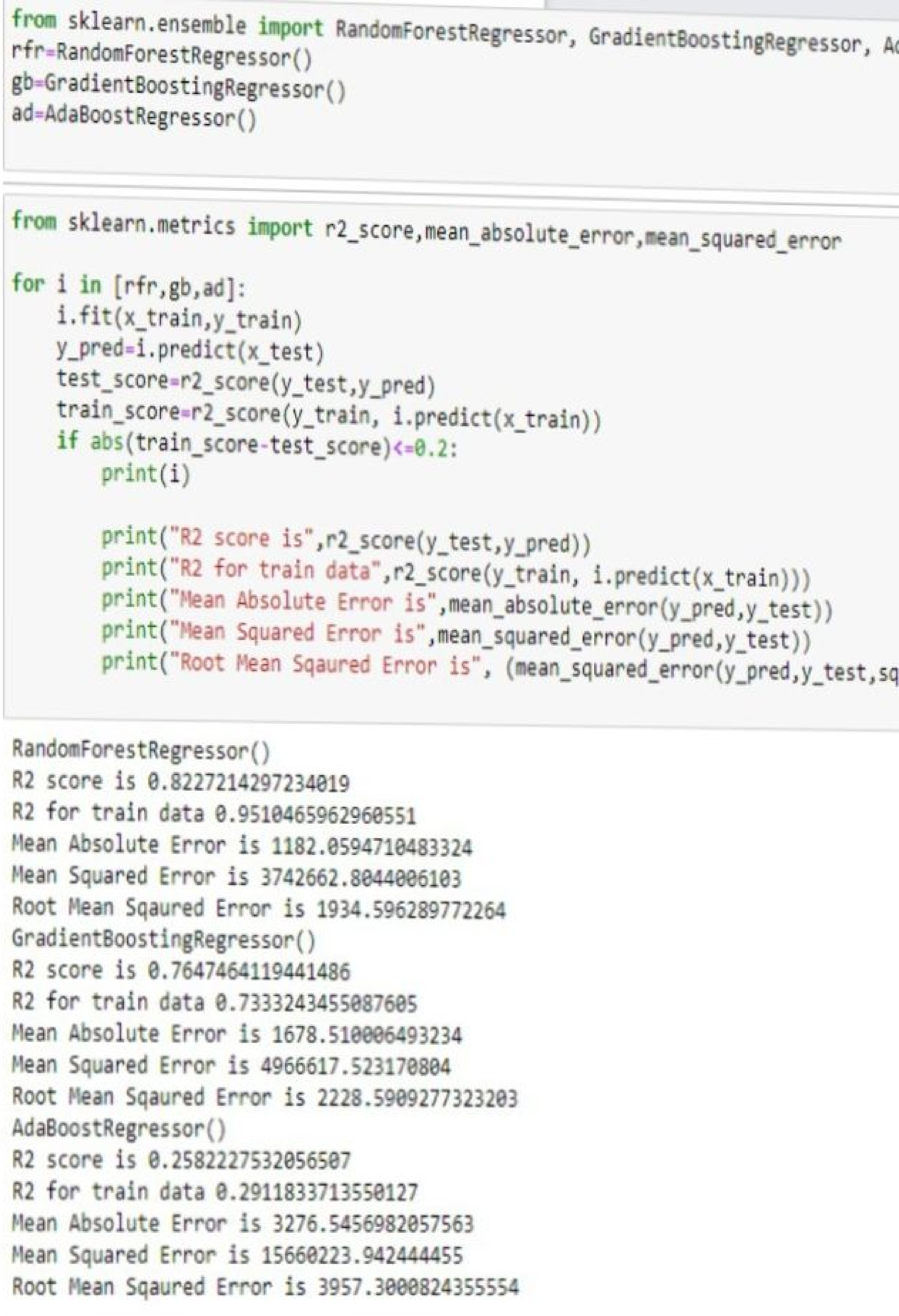


```
from sklearn.neighbors import KNeighborsRegresso
```



```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

knn=KNeighborsRegressor()
svr=SVR()
dt=DecisionTreeRegressor()

for i in [knn,svr,dt]:
    i.fit(x_train,y_train)
    y_pred=i.predict(x_test)
    test_score=r2_score(y_test,y_pred)
    train_score=r2_score(y_train,i.predict(x_train))
    if abs(train_score-test_score)<=0.1:
        print(i)
        print('R2 Score is',r2_score(y_test,y_pred))
        print('R2 Score for train data',r2_score(y_train,i.predict(x_train)))
        print('Mean Absolute Error is',mean_absolute_error(y_test,y_pred))
        print('Mean Squared Error is',mean_squared_error(y_test,y_pred))
        print('Root Mean Squared Error is',(mean_squared_error(y_test,y_pred,squ
```

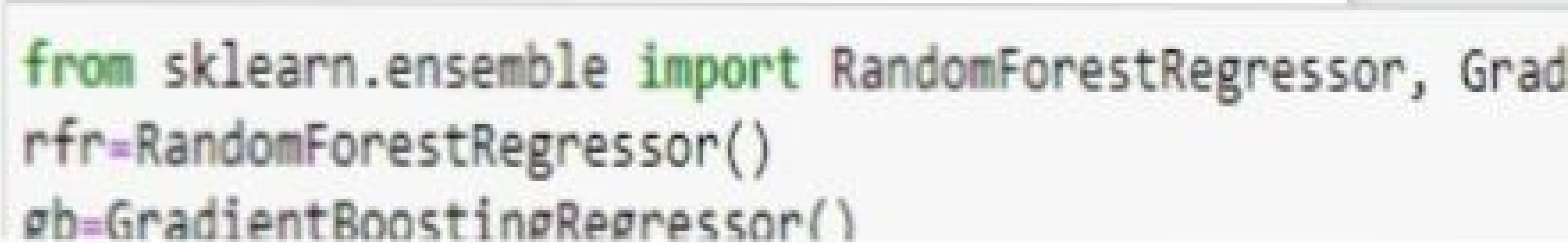
```
KNeighborsRegressor()
R2 Score is 0.7354576039734038
R2 Score for train data 0.7910150823510993
Mean Absolute Error is 1635.3106223678053
Mean Squared Error is 5584955.836743098
Root Mean Squared Error is 2363.2511158874117
SVR()
R2 Score is -0.007934481035057894
R2 Score for train data -0.012381130959185693
Mean Absolute Error is 3631.923243955232
Mean Squared Error is 21279271.857602067
Root Mean Squared Error is 4612.94611475162
```



```
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, Ad
rfr=RandomForestRegressor()
gb=GradientBoostingRegressor()
ad=AdaBoostRegressor()
```

```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

for i in [rfr,gb,ad]:
    i.fit(x_train,y_train)
    y_pred=i.predict(x_test)
    test_score=r2_score(y_test,y_pred)
    train_score=r2_score(y_train, i.predict(x_train))
    if abs(train_score-test_score)<=0.2:
        print(i)

        print("R2 score is",r2_score(y_test,y_pred))
        print("R2 for train data",r2_score(y_train, i.predict(x_train)))
        print("Mean Absolute Error is",mean_absolute_error(y_pred,y_test))
        print("Mean Squared Error is",mean_squared_error(y_pred,y_test))
        print("Root Mean Sqaured Error is", (mean_squared_error(y_pred,y_test,sq
```

```
RandomForestRegressor()
R2 score is 0.8227214297234019
R2 for train data 0.9510465962960551
Mean Absolute Error is 1182.0594710483324
Mean Squared Error is 3742662.8044006103
Root Mean Sqaured Error is 1934.596289772264
GradientBoostingRegressor()
R2 score is 0.7647464119441486
R2 for train data 0.7333243455087605
Mean Absolute Error is 1678.510006493234
Mean Squared Error is 4966617.523170804
Root Mean Sqaured Error is 2228.5909277323203
AdaBoostRegressor()
R2 score is 0.2582227532056507
R2 for train data 0.2911833713550127
Mean Absolute Error is 3276.5456982057563
Mean Squared Error is 15660223.942444455
Root Mean Sqaured Error is 3957.3000824355554
```



```
from sklearn.ensemble import RandomForestRegressor, Grad
rfr=RandomForestRegressor()
gb=GradientBoostingRegressor()
```

# Model Building

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. for this project we are applying four regression algorithms. The best model is saved based on its performance.