Telemonitoring of Parkinson's disease progression

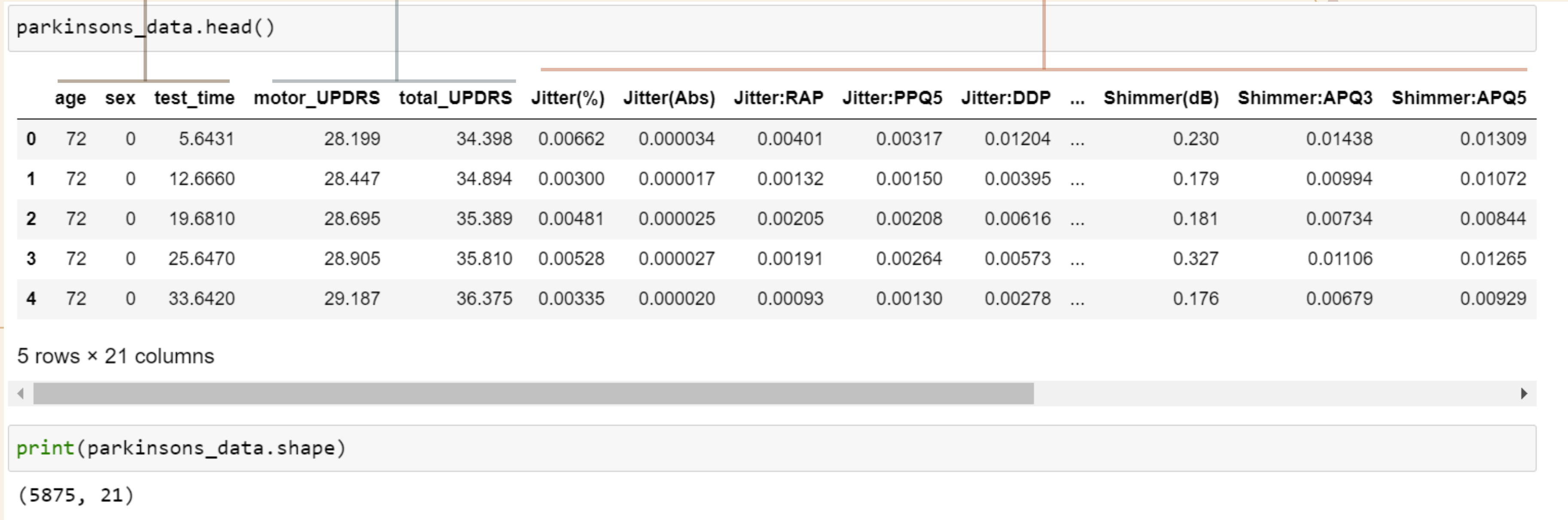
Comparing between machine learning techniques that can be used for improve the prediction of Parkinson's disease

About the dataset

Data of each patient

UPDRS

medical sound measurements



```
parkinsons_data.head()
```

	age	sex	test_time	motor_UPDRS	total_UPDRS	Jitter(%)	Jitter(Abs)	Jitter:RAP	Jitter:PPQ5	Jitter:DDP	...	Shimmer(dB)	Shimmer:APQ3	Shimmer:APQ5
0	72	0	5.6431	28.199	34.398	0.00662	0.000034	0.00401	0.00317	0.01204	...	0.230	0.01438	0.01309
1	72	0	12.6660	28.447	34.894	0.00300	0.000017	0.00132	0.00150	0.00395	...	0.179	0.00994	0.01072
2	72	0	19.6810	28.695	35.389	0.00481	0.000025	0.00205	0.00208	0.00616	...	0.181	0.00734	0.00844
3	72	0	25.6470	28.905	35.810	0.00528	0.000027	0.00191	0.00264	0.00573	...	0.327	0.01106	0.01265
4	72	0	33.6420	29.187	36.375	0.00335	0.000020	0.00093	0.00130	0.00278	...	0.176	0.00679	0.00929

5 rows × 21 columns

```
print(parkinsons_data.shape)
```

```
(5875, 21)
```

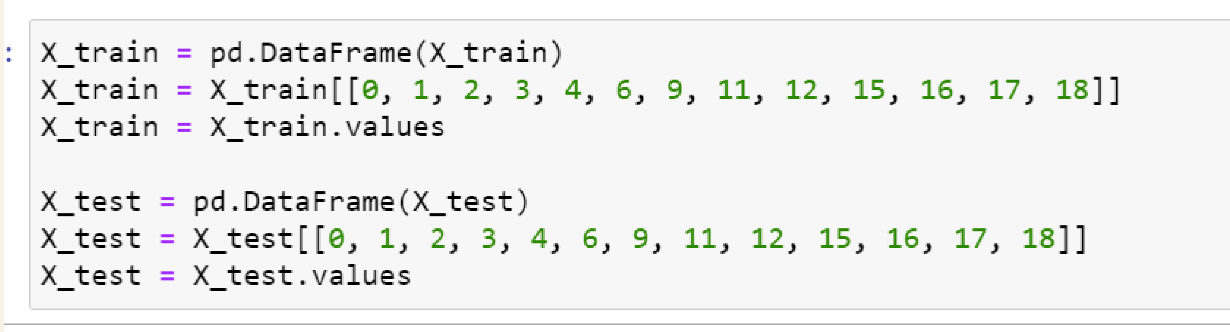
Feature selection

01 Filter Method

02 Wrapper Method

03 Embedded Method

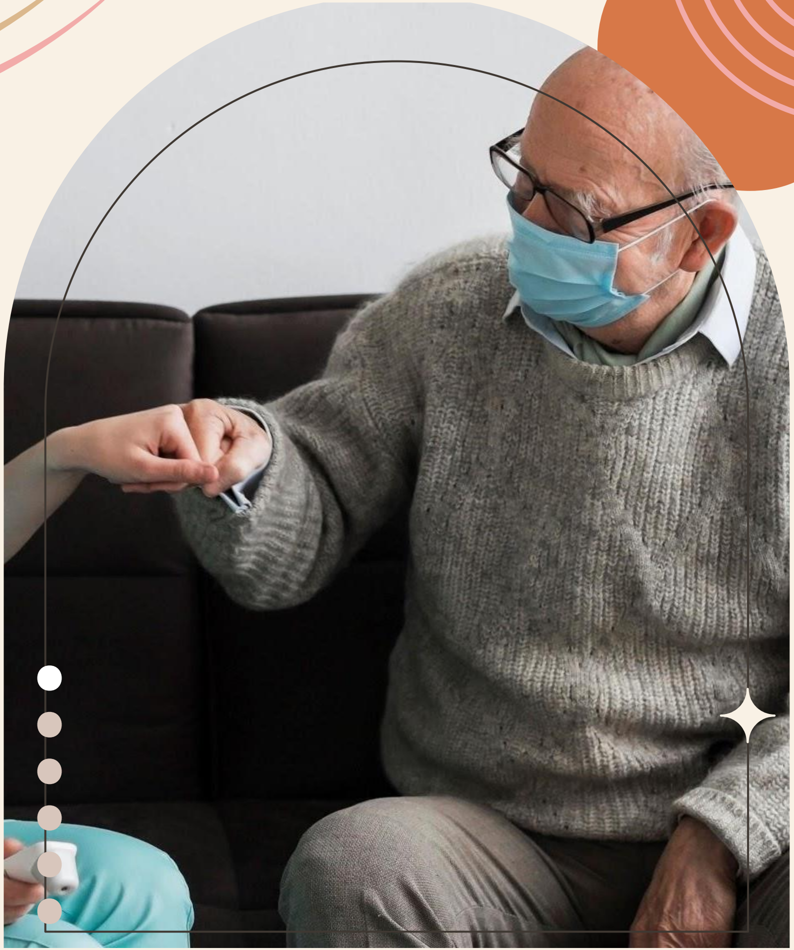
Backward Elimination



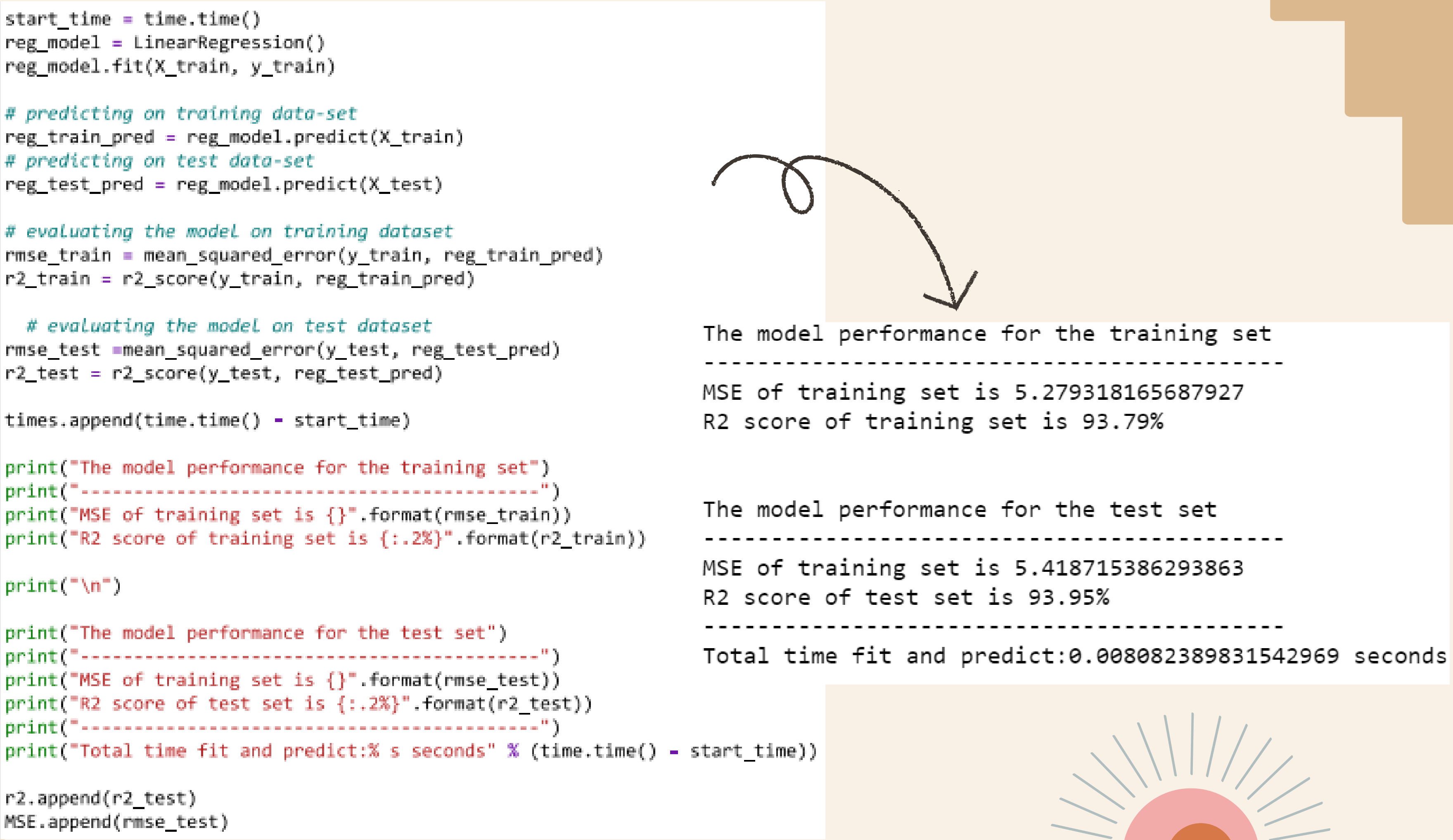
```
X_train = pd.DataFrame(X_train)
X_train = X_train[[0, 1, 2, 3, 4, 6, 9, 11, 12, 15, 16, 17, 18]]
X_train = X_train.values

X_test = pd.DataFrame(X_test)
X_test = X_test[[0, 1, 2, 3, 4, 6, 9, 11, 12, 15, 16, 17, 18]]
X_test = X_test.values
```

Model training

In this part we will use ML models, so they learn from our data and make predictions.



01 Linear Regression

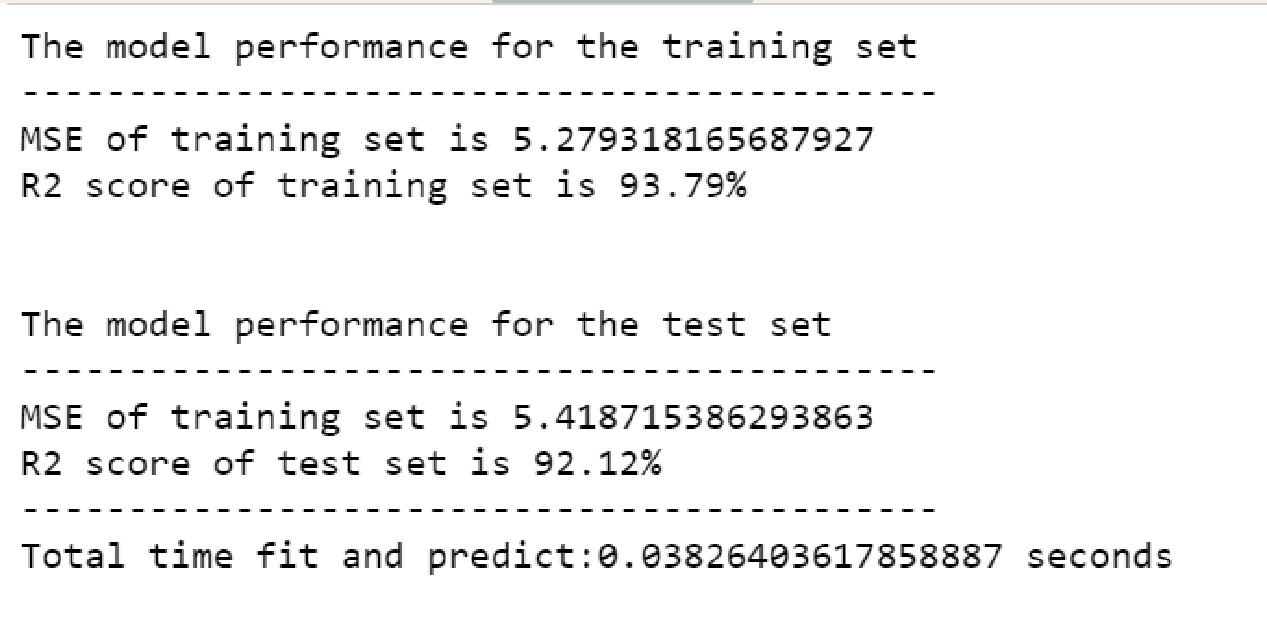


```python
start_time = time.time()
reg_model = LinearRegression()
reg_model.fit(X_train, y_train)

# predicting on training data-set
reg_train_pred = reg_model.predict(X_train)
# predicting on test data-set
reg_test_pred = reg_model.predict(X_test)

# evaluating the model on training dataset
rmse_train = mean_squared_error(y_train, reg_train_pred)
r2_train = r2_score(y_train, reg_train_pred)

  # evaluating the model on test dataset
rmse_test =mean_squared_error(y_test, reg_test_pred)
r2_test = r2_score(y_test, reg_test_pred)

times.append(time.time() - start_time)

print("The model performance for the training set")
print("-------------------------------------------")
print("MSE of training set is {}".format(rmse_train))
print("R2 score of training set is {:.2%}".format(r2_train))

print("\n")

print("The model performance for the test set")
print("-------------------------------------------")
print("MSE of training set is {}".format(rmse_test))
print("R2 score of test set is {:.2%}".format(r2_test))
print("-------------------------------------------")
print("Total time fit and predict:% s seconds" % (time.time() - start_time))

r2.append(r2_test)
MSE.append(rmse_test)
```

```
The model performance for the training set
-------------------------------------------
MSE of training set is 5.279318165687927
R2 score of training set is 93.79%


The model performance for the test set
-------------------------------------------
MSE of training set is 5.418715386293863
R2 score of test set is 93.95%
-------------------------------------------
Total time fit and predict:0.008082389831542969 seconds
```

02

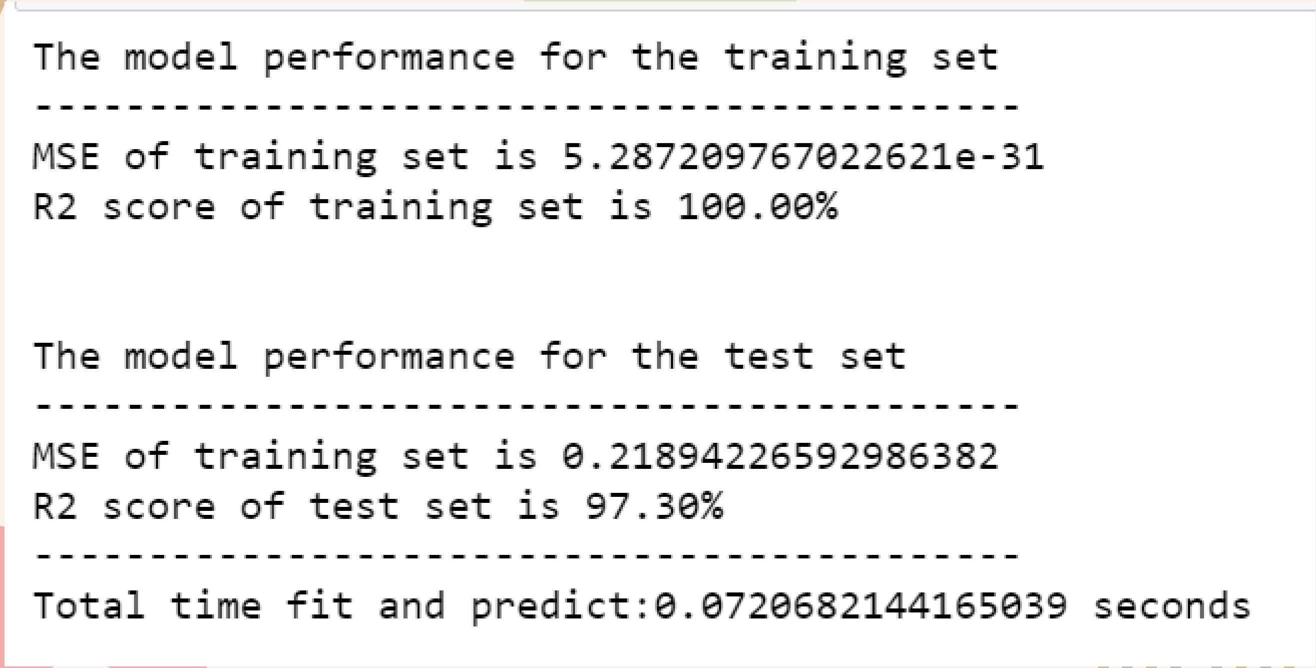
Polynomial Regression



```
The model performance for the training set
-------------------------------------------
MSE of training set is 5.279318165687927
R2 score of training set is 93.79%


The model performance for the test set
-------------------------------------------
MSE of training set is 5.418715386293863
R2 score of test set is 92.12%
-------------------------------------------
Total time fit and predict:0.03826403617858887 seconds
```

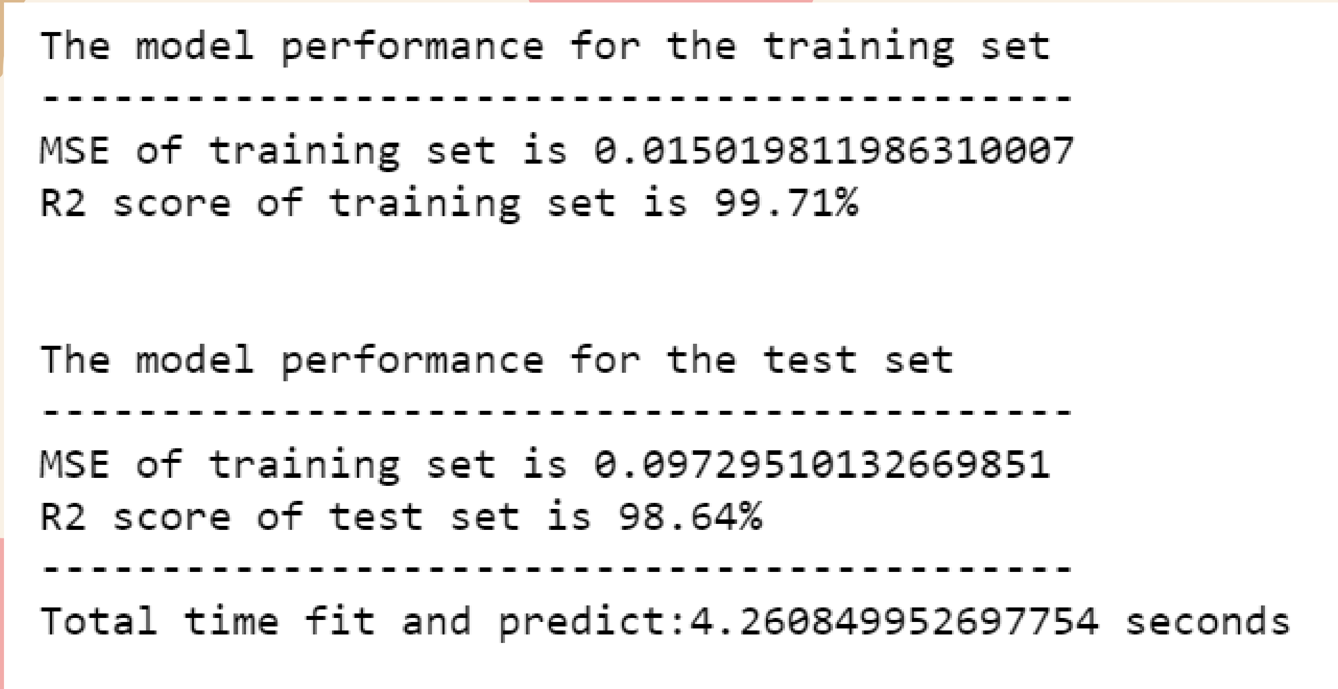
03

Decision Tree Regressor



```
The model performance for the training set
-------------------------------------------
MSE of training set is 5.287209767022621e-31
R2 score of training set is 100.00%


The model performance for the test set
-------------------------------------------
MSE of training set is 0.21894226592986382
R2 score of test set is 97.30%
-------------------------------------------
Total time fit and predict:0.0720682144165039 seconds
```

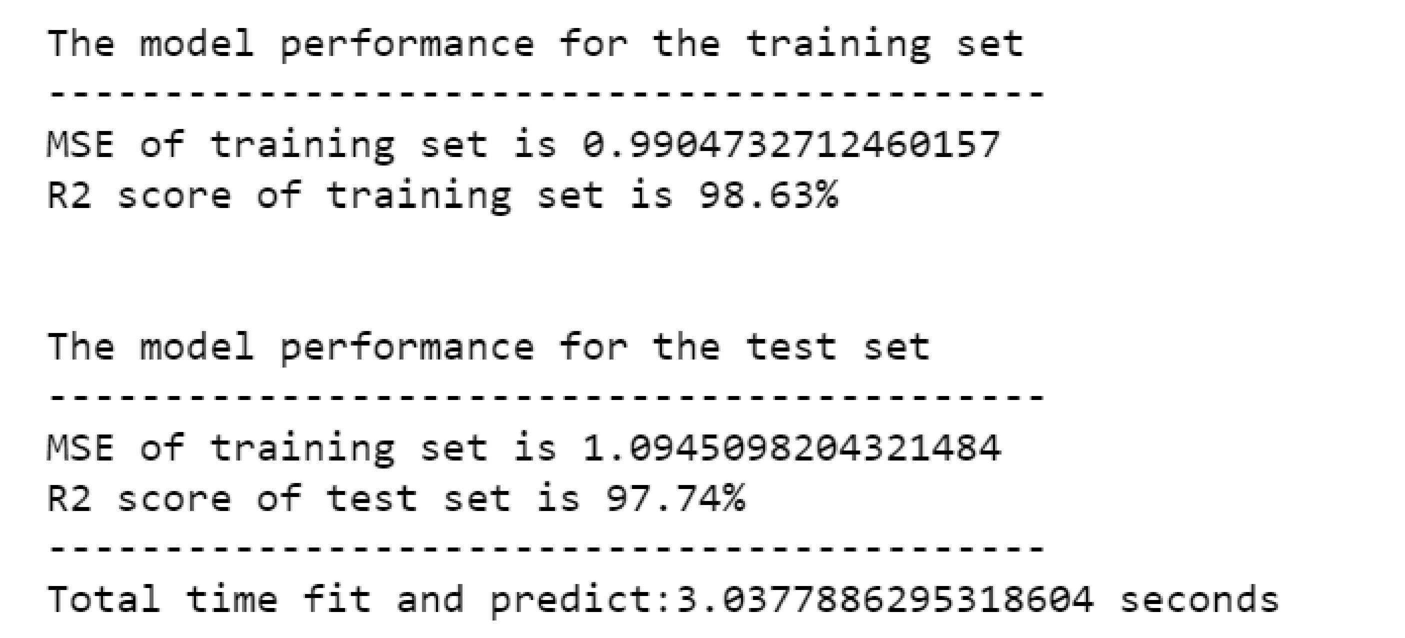
04
Random Forest



```
The model performance for the training set
-------------------------------------------
MSE of training set is 0.015019811986310007
R2 score of training set is 99.71%


The model performance for the test set
-------------------------------------------
MSE of training set is 0.09729510132669851
R2 score of test set is 98.64%
-------------------------------------------
Total time fit and predict:4.260849952697754 seconds
```

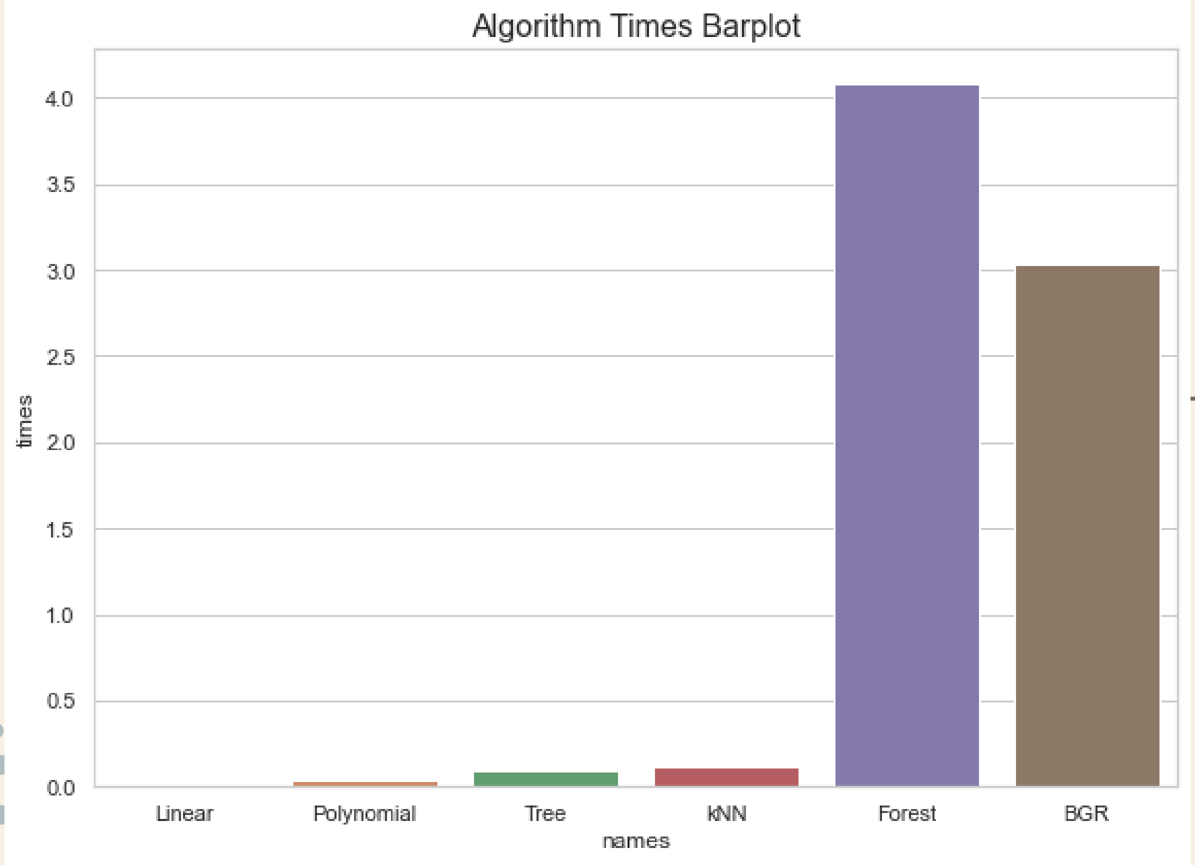
05

Gradient Boosting Regressor



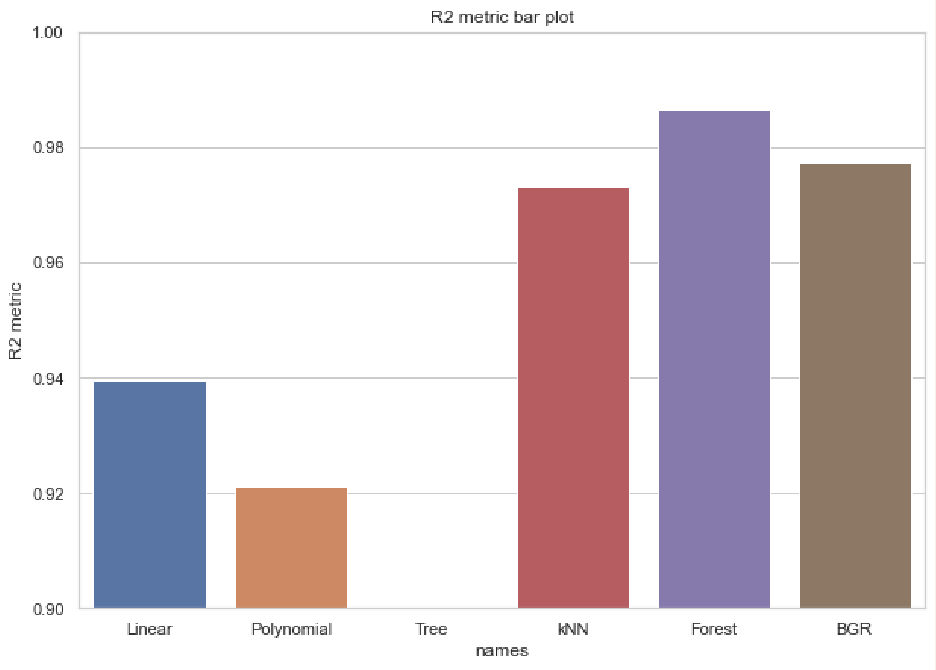
```
The model performance for the training set
-------------------------------------------
MSE of training set is 0.9904732712460157
R2 score of training set is 98.63%


The model performance for the test set
-------------------------------------------
MSE of training set is 1.0945098204321484
R2 score of test set is 97.74%
-------------------------------------------
Total time fit and predict:3.0377886295318604 seconds
```

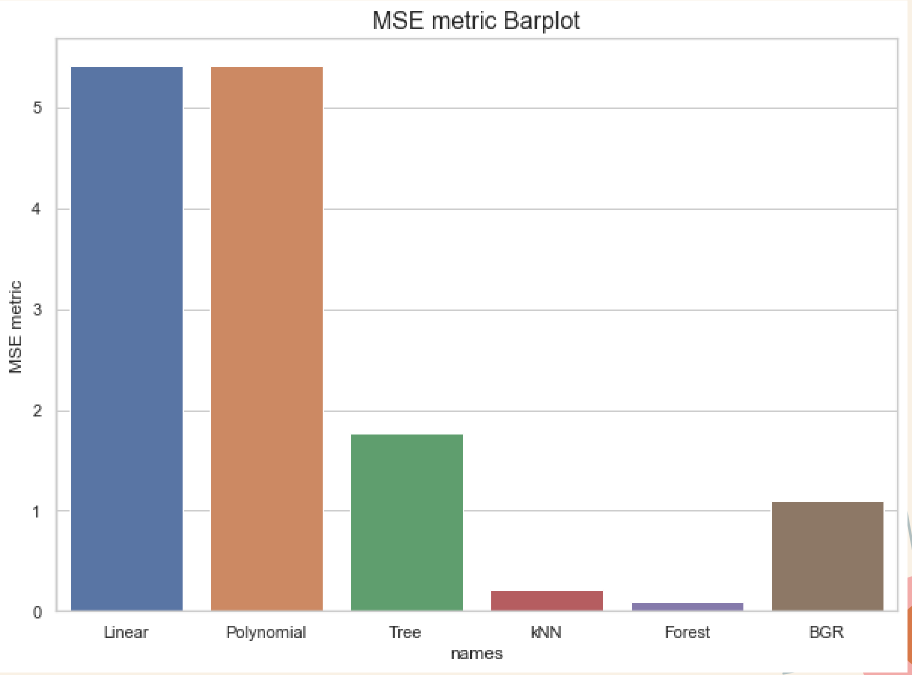
Time Plots



we compare the times that each of the algorithms runs, We see that the forest need longer

Matrix Plots





we see that forests have the best possible performance.

Thanks for listining

Rahaf Alghamdi

Online Data Science Bootcamps for SDAIA Academy