Aayush Shah
19BCE245
23 September 2022

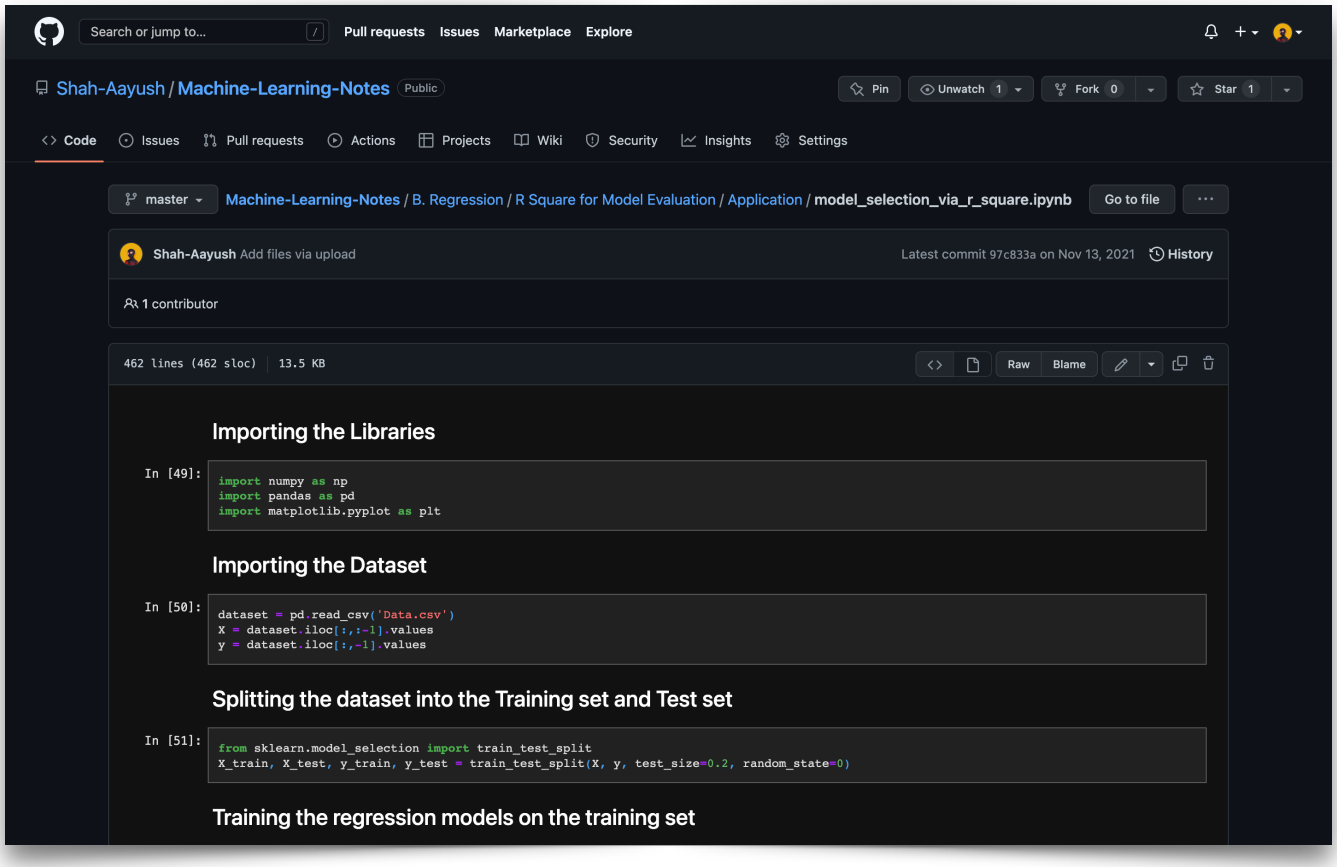
Big Data Analytics

Practical 2

Aim

Learning limitation of data analytics by applying Machine Learning Techniques on large amount of data. Write a program to read data set from any online website, excel file and CSV file and to perform Linear regression and logistic regression on iris dataset

Various regressions applied on random dataset



1. Decision Tree Regression

In [52]:
```
from sklearn.tree import DecisionTreeRegressor
decisionTree_regressor = DecisionTreeRegressor(random_state=0)
decisionTree_regressor.fit(X_train, y_train)
```

Out[52]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=0, splitter='best')
```

In [53]:
```
y_pred_decisionTree = decisionTree_regressor.predict(X_test)
```

2. Multiple Linear Regression

In [54]:
```
from sklearn.linear_model import LinearRegression
linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)
```

Out[54]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [55]:
```
y_pred_multipleLinear = linear_regressor.predict(X_test)
```

3. Polynomial Regression

In [56]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
poly_reg = PolynomialFeatures(degree=4)   # set degree
X_poly = poly_reg.fit_transform(X_train)
polynomial_regressor = LinearRegression()
polynomial_regressor.fit(X_poly, y_train)
```

Out[56]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [57]:
```
y_pred_polynomial = polynomial_regressor.predict(poly_reg.transform(X_test))
```

4. Support Vector Regression (SVR)

In [58]:
```
# Preprocessing (Feature Scaling) and reshaping :
y_svr = dataset.iloc[:,-1].values.reshape(len(y), 1)
X_train_svr, X_test_svr, y_train_svr, y_test_svr = train_test_split(X, y_svr, test_size = 0.2, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X_train_svr = sc_X.fit_transform(X_train_svr)
y_train_svr = sc_y.fit_transform(y_train_svr)
```

In [59]:
```
from sklearn.svm import SVR
svr_regressor = SVR(kernel = 'rbf')
svr_regressor.fit(X_train_svr, y_train_svr)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:760: DataConversionWarning: A column-vector y was passed when a 1d arra
y was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[59]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [60]:
```
y_pred_svr = sc_y.inverse_transform(svr_regressor.predict(sc_X.transform(X_test_svr)))
```

5. Random Forest Regression

In [61]:
```
from sklearn.ensemble import RandomForestRegressor
randomForest_regressor = RandomForestRegressor(n_estimators=10, random_state=0)
randomForest_regressor.fit(X_train, y_train)
```

Out[61]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=10, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

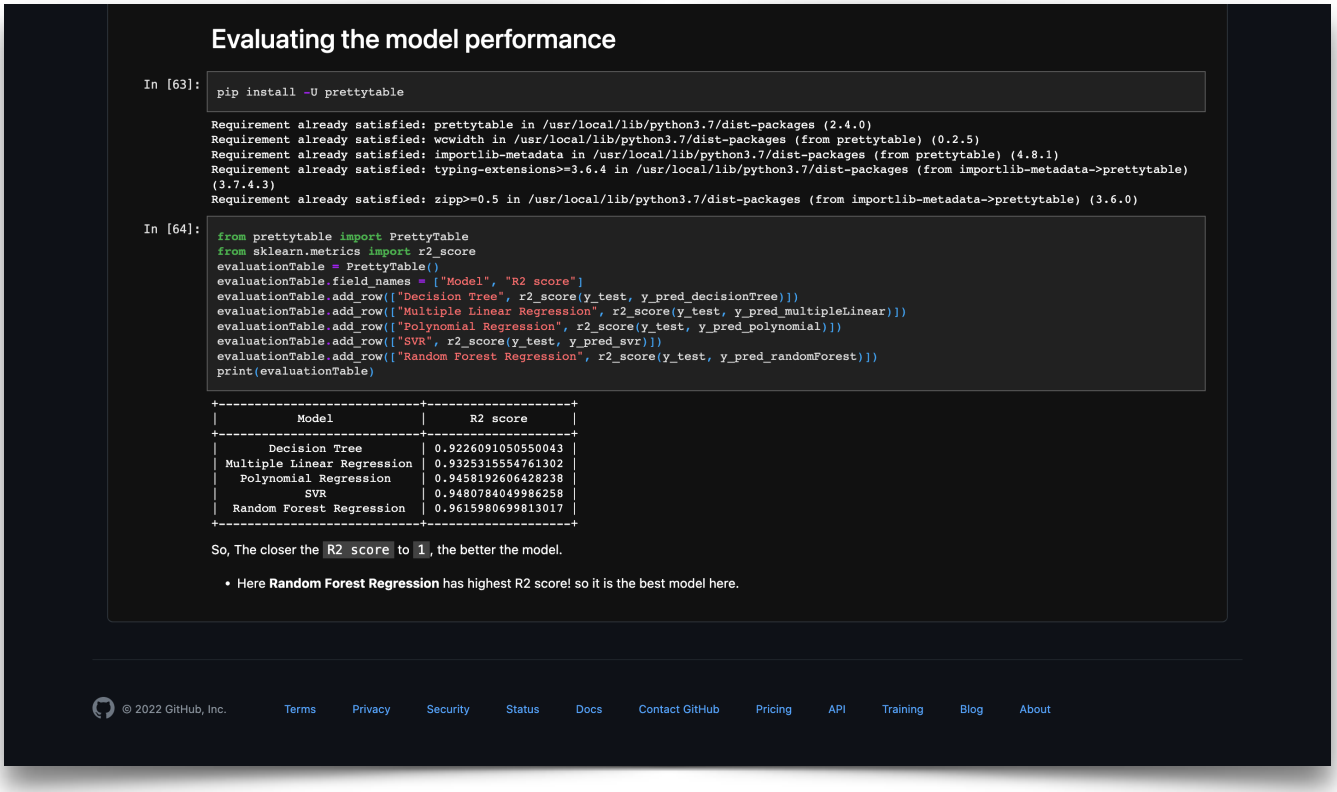
In [62]:
```
y_pred_randomForest = randomForest_regressor.predict(X_test)
```

Evaluating the model performance

In [63]:
```
pip install -U prettytable
```

```
Requirement already satisfied: prettytable in /usr/local/lib/python3.7/dist-packages (2.4.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages (from prettytable) (0.2.5)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from prettytable) (4.8.1)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->prettytable)
(3.7.4.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->prettytable) (3.6.0)
```

In [64]:
```
from prettytable import PrettyTable
from sklearn.metrics import r2_score
evaluationTable = PrettyTable()
evaluationTable.field_names = ["Model", "R2 score"]
evaluationTable.add_row(["Decision Tree", r2_score(y_test, y_pred_decisionTree)])
evaluationTable.add_row(["Multiple Linear Regression", r2_score(y_test, y_pred_multipleLinear)])
evaluationTable.add_row(["Polynomial Regression", r2_score(y_test, y_pred_polynomial)])
```



Evaluating the model performance

In [63]:
```
pip install -U prettytable
```

```
Requirement already satisfied: prettytable in /usr/local/lib/python3.7/dist-packages (2.4.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages (from prettytable) (0.2.5)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from prettytable) (4.8.1)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->prettytable)
(3.7.4.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->prettytable) (3.6.0)
```

In [64]:
```
from prettytable import PrettyTable
from sklearn.metrics import r2_score
evaluationTable = PrettyTable()
evaluationTable.field_names = ["Model", "R2 score"]
evaluationTable.add_row(["Decision Tree", r2_score(y_test, y_pred_decisionTree)])
evaluationTable.add_row(["Multiple Linear Regression", r2_score(y_test, y_pred_multipleLinear)])
evaluationTable.add_row(["Polynomial Regression", r2_score(y_test, y_pred_polynomial)])
evaluationTable.add_row(["SVR", r2_score(y_test, y_pred_svr)])
evaluationTable.add_row(["Random Forest Regression", r2_score(y_test, y_pred_randomForest)])
print(evaluationTable)
```

```
+-----------------------------+--------------------+
|            Model            |      R2 score      |
+-----------------------------+--------------------+
|       Decision Tree         | 0.9226091050550043 |
| Multiple Linear Regression  | 0.9325315554761302 |
|   Polynomial Regression     | 0.9458192606428238 |
|            SVR              | 0.9480784049986258 |
|  Random Forest Regression   | 0.9615980699813017 |
+-----------------------------+--------------------+
```

So, The closer the `R2 score` to `1`, the better the model.

- Here **Random Forest Regression** has highest R2 score! so it is the best model here.

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Conclusion

From this practical, we gain insights about why big data analytics is needed if we want to perform regressions or any techniques on larger datasets because simple machine learning techniques are not useful in these cases.