WEEK- 7: Explore other regression algorithm.

- Other Regression algorithm such as

1. Support Vector Regression
2. Lasso Regression
3. Random Forest

Session No.1

In Machine Learning, we use various kinds of algorithms to allow machines to learn the relationships within the data provided and make predictions based on patterns or rules identified from the dataset. So, regression is a machine learning technique where the model predicts the output as a continuous numerical value.

Source: https://www.hindish.com

Regression analysis is often used in finance, investing, and others, and finds out the relationship between a single dependent variable(target variable) dependent on several independent ones. For example, predicting house price, stock market or salary of an employee, etc are the most common regression problems.

The algorithms we are going to cover are:

1. Linear Regression

2. Decision Tree

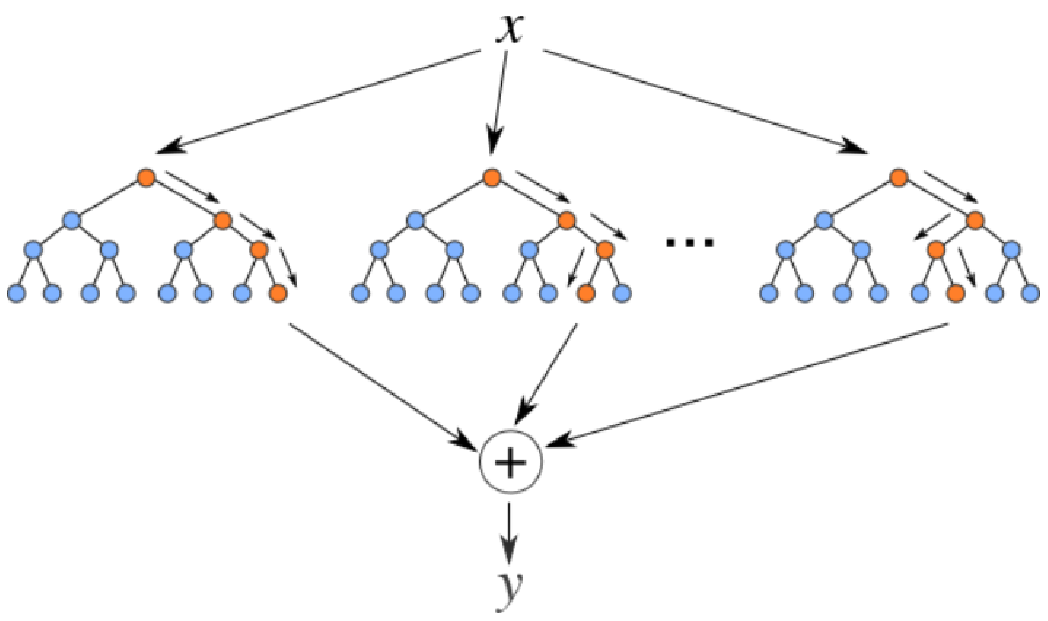
3. Support Vector Regression

4. Lasso Regression

5. Random Forest

1.Random Forest Regressor

Random Forests are an ensemble(combination) of decision trees. It is a Supervised Learning algorithm used for classification and regression. The input data is passed through multiple decision trees. It executes by constructing a different number of decision trees at training time and outputting the class that is the mode of the classes (for classification) or mean prediction (for regression) of the individual trees.



Source: https://levelup.gitconnected.com

Pros:

- Good at learning complex and non-linear relationships
- Very easy to interpret and understand

Cons:

- They are prone to overfitting

- Using larger random forest ensembles to achieve higher performance slows down their speed and then they also need more memory.

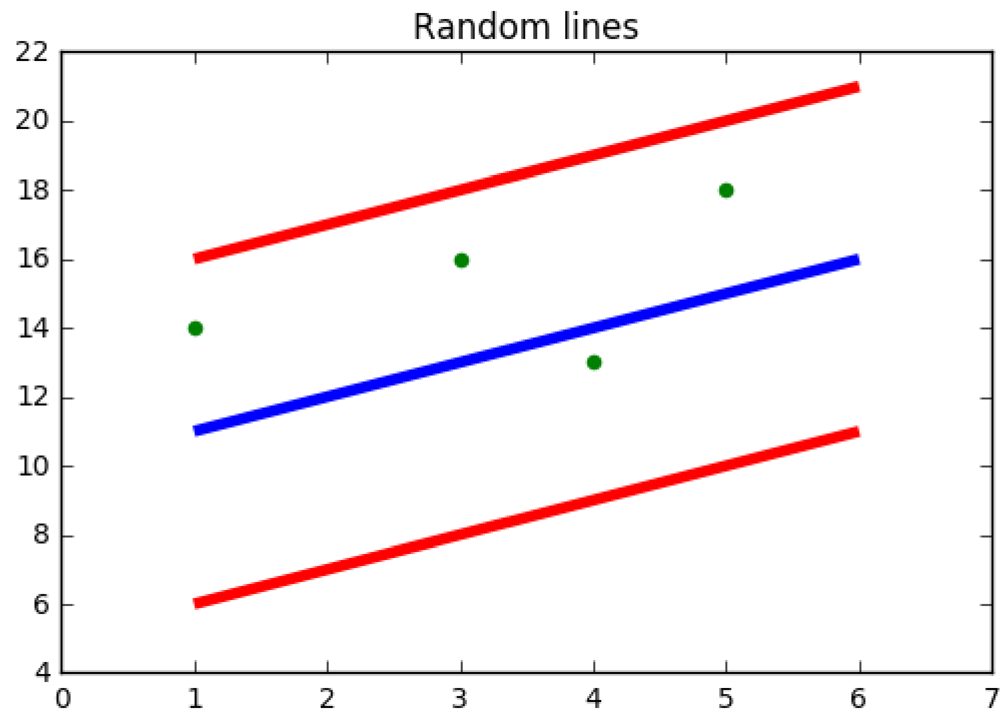
Implementation

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
X, y = make_regression(n_features=4, n_informative=2, random_state=0, shuffle=False)
rfr = RandomForestRegressor(max_depth=3)
rfr.fit(X, y)
print(rfr.predict([[0, 1, 0, 1]]))
```

Output:

2.Support Vector Regression

You must have heard about SVM i.e., Support Vector Machine. SVR also uses the same idea of SVM but here it tries to predict the real values. This algorithm uses hyperplanes to segregate the data. In case this separation is not possible then it uses kernel trick where the dimension is increased and then the data points become separable by a hyperplane.



Source: https://www.medium.com

In the figure above, **the Blue line is the Hyper Plane; Red Line is the Boundary Line**All the data points are within the boundary line(Red Line). The main objective of SVR is to basically consider the points that are within the boundary line.

Pros:

- Robust to outliers.
- Excellent generalization capability
- High prediction accuracy.

Cons:

- Not suitable for large datasets.
- They do not perform very well when the data set has more noise.

Implementation

```
from sklearn.svm import SVR
import numpy as np
rng = np.random.RandomState(1)
```

```
X = np.sort(5 * rng.rand(80, 1), axis=0)

y = np.sin(X).ravel()

y[::5] += 3 * (0.5 - rng.rand(16))

# Fit regression model

svr = SVR().fit(X, y)

# Predict

X_test = np.arange(0.0, 5.0, 1)[:, np.newaxis]

svr.predict(X_test)
```

Output:

```
array([-0.07840308,  0.78077042,  0.81326895,  0.08638149, -0.6928019 ])
```

3. Lasso Regression

- LASSO stands for Least Absolute Selection Shrinkage Operator. Shrinkage is basically defined as a constraint on attributes or parameters.
- The algorithm operates by finding and applying a constraint on the model attributes that cause regression coefficients for some variables to shrink toward a zero.
- Variables with a regression coefficient of zero are excluded from the model.
- So, lasso regression analysis is basically a shrinkage and variable selection method and it helps to determine which of the predictors are most important.

Pros:

- It avoids overfitting

Cons:

- LASSO will select only one feature from a group of correlated features
- Selected features can be highly biased.

Implementation

```
from sklearn import linear_model

import numpy as np

rng = np.random.RandomState(1)

X = np.sort(5 * rng.rand(80, 1), axis=0)

y = np.sin(X).ravel()
```

```
y[::5] += 3 * (0.5 - rng.rand(16))
# Fit regression model
lassoReg = linear_model.Lasso(alpha=0.1)
lassoReg.fit(X,y)
# Predict
X_test = np.arange(0.0, 5.0, 1)[:, np.newaxis]
lassoReg.predict(X_test)
```

Output: