Random-Forest-Regression

January 14, 2025

[ ]:
```
'''
   Random Forest Regression -->

   Random Forest is an ensemble learning method that combines multiple
   decision trees to improve prediction accuracy and reduce overfitting.
   It can be used for both regression and classification tasks.

   Ensemble Learning -->

   Ensemble learning is a machine learning technique where multiple models
   (often called "weak learners" or "base models") are combined to solve a␣
↪problem
   and improve overall performance. The primary goal is to leverage the␣
↪strengths of
   individual models and reduce their weaknesses by aggregating their␣
↪predictions.

   Key Concepts of Ensemble Learning -->

   Diversity:

   The base models in an ensemble should make different types of errors to
   achieve better overall performance when combined.

   Aggregation:

   Predictions from the individual models are combined, often through methods
   like averaging (for regression) or voting (for classification).

   Generalization:

   Ensembles are designed to generalize better on unseen data by reducing
   overfitting and variance.
'''
```

[ ]:
```
'''
   Types of Ensemble Learning -->
```

```
   Bagging (Bootstrap Aggregating):

   Combines predictions from multiple models trained on different subsets of␣
↪the training data.
   Example: Random Forest.
   Reduces variance by averaging results.

   Boosting:

   Builds models sequentially, where each model tries to correct the errors of␣
↪its predecessor.
   Examples: AdaBoost, Gradient Boosting, XGBoost.
   Reduces bias and variance.

   Stacking:

   Combines predictions from multiple models using a "meta-model" that learns␣
↪to
   weight the predictions of the base models.
   Example: A neural network or logistic regression used as a meta-model.

   Voting/Blending:

   Combines the predictions of multiple models by majority vote
   (classification) or averaging (regression).
'''
```

[ ]:
```
'''
   Advantages -->

   Improved Accuracy: Combines strengths of multiple models.
   Reduced Overfitting: By aggregating predictions, ensembles are less likely␣
↪to
   overfit compared to individual models.
   Versatility: Works with different types of base models and algorithms.

   Disadvantages -->

   Increased Complexity: More computationally intensive and harder to␣
↪interpret.
   Training Time: Longer training times compared to individual models.
   Risk of Redundancy: If base models are too similar, the benefits of␣
↪diversity are lost.

   Applications -->
```

```
    Spam detection
    Fraud detection
    Image classification
    Predictive modeling tasks in finance, healthcare, and more
'''
```

[1]:
```
#   Importing Libraries -->

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
```

[2]:
```
#   Importing Dataset -->

data = pd.read_csv('Data/Position_Salaries.csv')
data
```

[2]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

[3]:
```
x_data = data.iloc[:, 1:-1].values
y_data = data.iloc[:, -1].values
```

[6]:
```
#   Building Model -->

model = RandomForestRegressor(n_estimators = 10, random_state = 0)
model.fit(x_data, y_data)
```

[6]:
```
RandomForestRegressor(n_estimators=10, random_state=0)
```

[7]:
```
#   Predicting Result -->

y_pred = model.predict([[6.5]])
y_pred
```

[7]:
```
array([167000.])
```

[8]:
```
#   Visualizing The Results -->

x_grid = np.arange(min(x_data), max(x_data), 0.1)
x_grid = x_grid.reshape((len(x_grid), 1))
plt.scatter(x_data, y_data, color = 'red')
plt.plot(x_grid, model.predict(x_grid), color = 'blue')
plt.title("Random Forest Regression")
plt.show()
```

```
C:\Users\krish\AppData\Local\Temp\ipykernel_17248\2088354927.py:3:
DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is
deprecated, and will error in future. Ensure you extract a single element from
your array before performing this operation. (Deprecated NumPy 1.25.)
  x_grid = np.arange(min(x_data), max(x_data), 0.1)
```