**SUPPORT VECTOR REGRESSION**

**CODE EXPLANATION:**

SVR is a type of regression analysis that uses the support vector machine (SVM) algorithm to predict a continuous variable, in this case, salary based on position levels.

1. **Connecting to the google drive:** This code block imports the Google Drive API to access files stored in Google Drive. The command "drive.mount" mounts the drive, and the "/content/drive" specifies the path to the Drive.
2. **Importing the libraries:** This code block imports the required modules for machine learning, including numpy, pandas, and matplotlib.
3. **Importing the dataset:** This code block imports the dataset using pandas' "read\_csv" function and assigns the third and fourth columns of the dataset to variables x and y, respectively. It also prints the dataset, x, and y to confirm the data import.
4. **Feature Scaling:** This code block scales the features in x and y to ensure they are on the same scale. It uses the StandardScaler() function from Scikit-learn to scale the data.
5. **Training the model:** This code block creates an instance of the Support Vector Regression (SVR) model, sets the kernel parameter to 'rbf' (Radial Basis Function), and fits the model to the scaled data using the "fit" method.
6. **Predicting the result:** This code block predicts the salary of an employee with position level 6.5 using the "predict" method. It first scales the input using the "transform" method of the StandardScaler object created in the Feature Scaling block, then takes the inverse of the scaled value to go back to the initial scale.
7. **Visualizing the Results:** This code block plots the original data points (in red) and the predicted values (in blue) using the "scatter" and "plot" functions of matplotlib. The title, x-axis label, and y-axis label are added using the corresponding functions. The second plot adds a smoothed curve by setting the interval between values on the x-axis to 0.01 using the "np.arange" function, plotting the values on the curve using the "plot" function, and displaying the plot using "show".

**RANDOM FOREST REGRESSION**

**CODE EXPLANATION:**

Random forest regression is a machine learning algorithm used for predicting numerical data, such as prices or stock prices. It works by creating many decision trees at training time and outputting the average of the trees' predictions. Each decision tree is trained on a subset of the data and a subset of the features, which helps to reduce overfitting and improve accuracy. Random forest regression can handle both categorical and continuous data and is relatively robust to outliers and noise. It can be used for both classification and regression tasks and is a popular algorithm in machine learning due to its high accuracy and ease of implementation.

1. **Connecting to the Google Drive:** In this code block, the Google Drive is connected to access the data file. It imports the drive module from the google.colab library and mounts the drive to the /content/drive directory using the drive.mount() method.
2. **Importing the Libraries:** This code block imports the required Python libraries/modules like numpy, pandas, and matplotlib. numpy is used for numerical computing, pandas is used for data manipulation and analysis, and matplotlib is used for data visualization.
3. **Importing the Dataset:** This code block imports the dataset from the Google Drive using pandas module's read\_csv() method. The iloc() method is then used to select the 2nd column to the second last column (excluding the last column) and assign it to the variable x. Similarly, the last column is assigned to the variable y.
4. **Training the Model:** In this code block, the RandomForestRegressor algorithm is used for regression. First, the RandomForestRegressor object is created with n\_estimators and random\_state hyperparameters. Then, the model is trained on the x and y data using the fit() method.
5. **Predicting the Result:** This code block uses the trained model to predict the salary for a given position level. The predict() method is used to predict the salary for a position level of 6.5. The predicted salary is stored in the Y\_pred variable, which is then printed to the console.
6. **Visualizing the Results:** In this code block, the data is visualized using matplotlib. First, the original data is plotted as a scatter plot with the position level on the x-axis and salary on the y-axis. Then, the predicted data is plotted as a line plot on the same graph. The title(), xlabel(), and ylabel() methods are used to set the title and labels for the plot. Finally, the plot is displayed using the show() method. The same process is repeated with a more fine-grained X\_grid variable to obtain a smoother plot of the regression line.

**DECISION TREE REGRESSION**

**CODE EXPLANATION:**

Decision tree regression is a type of regression analysis technique that predicts the value of a dependent variable by learning simple decision rules from the independent variables. In a decision tree regression, the data is split into several segments based on the values of the independent variables.

1. **Connecting to the google drive:** This code block connects the Google Colab notebook to Google Drive, which is where the file is stored. This is done by importing the drive module from the google.colab package and calling the mount() function with the argument '/content/drive', which mounts the Google Drive to the Colab environment.
2. **Importing the libraries:** This code block imports the necessary libraries for machine learning. It imports numpy as np, pandas as pd, and matplotlib.pyplot as plt.
3. **Importing the dataset:** This code block imports the dataset from Google Drive using the read\_csv() function from pandas. It reads in the file 'Position\_Salaries\_poly-1.csv' and assigns the second to second-to-last columns to x and the last column to y. It then prints out the entire dataset, x, and y for inspection.
4. **Training the model:** This code block creates and fits a DecisionTreeRegressor model to the dataset. First, it imports the DecisionTreeRegressor from sklearn.tree. It then creates a new DecisionTreeRegressor object called regressor with random\_state = 0. Finally, it calls the fit() method of regressor with x and y as arguments to train the model.
5. **Predicting the result:** This code block uses the trained model to predict the salary for a specific position level, 6.5. It does this by calling the predict() method of regressor with a new NumPy array containing the value 6.5. The result is stored in Y\_pred and printed out.
6. **Visualizing the Results:** This code block creates a scatter plot of the original data points in red, and the predicted values in blue. It first plots the original data points with plt.scatter(), and then plots the predicted values using the predict() method of regressor. The title, x-axis label, and y-axis label are added using plt.title(), plt.xlabel(), and plt.ylabel(), respectively. Finally, the plot is displayed using plt.show(). The same plot is then created again using a X\_grid array to create a smoother line between the predicted values.