Practicle 1:

In [7]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

In [8]:
```
# Load the dataset
data = pd.read_csv('Salary_Data.csv')
```

In [9]:
```
# Split the data into input (X) and output (y) variables
X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values
```

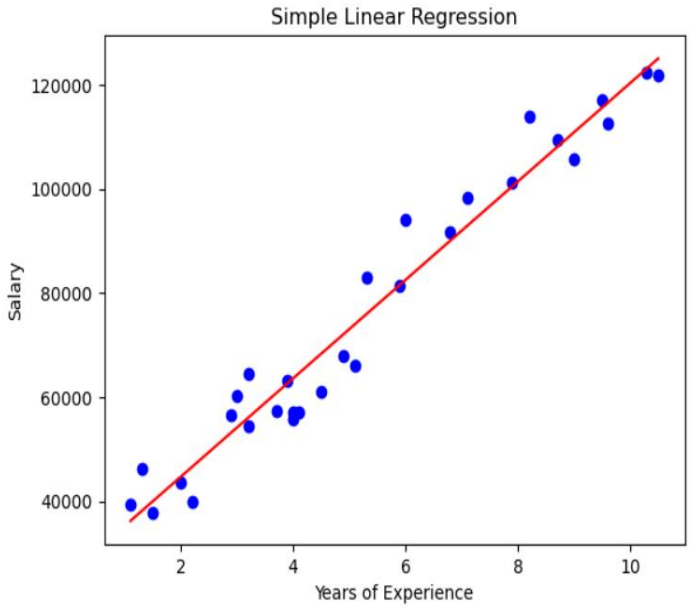
In [10]:
```
# Create a linear regression model
regressor = LinearRegression()
```

In [17]:
```
# Train the model
regressor.fit(X, y)
```

Out[17]:
```
LinearRegression
LinearRegression()
```

In [12]:
```
# Predict the salaries based on the model
y_pred = regressor.predict(X)
```

In [13]:
```
# Visualize the data and regression line
plt.scatter(X, y, color='blue')
plt.plot(X, y_pred, color='red')
plt.title('Simple Linear Regression')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```



In [14]:
```
# Print the coefficients and intercept
coefficients = regressor.coef_
intercept = regressor.intercept_
print('Coefficients:', coefficients)
print('Intercept:', intercept)
```

```
Coefficients: [9449.96232146]
Intercept: 25792.200198668717
```

Practical 2:

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
```

In [2]:
```
# Load the dataset
data = pd.read_csv('Salary_Data.csv')
```

In [3]:
```
# Split the dataset into features (X) and target variable (y)
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
```

In [4]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [5]:
```
# Create a KNN regression model
knn = KNeighborsRegressor(n_neighbors=5)
```

In [6]:
```
# Train the model
knn.fit(X_train, y_train)
```

Out[6]:
```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

In [7]:
```
# Predict the target variable for the test set
y_pred = knn.predict(X_test)
```

In [8]:
```
# Calculate the mean squared error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 50357290.43333333
```

In [9]:
```
# Print the predicted and actual values for the test set
print("Predicted\tActual")
for i in range(len(y_pred)):
    print(f"{y_pred[i]}\t\t{y_test[i]}")
```

```
Predicted       Actual
115249.0                112635.0
59394.4         67938.0
106311.4                113812.0
71904.8         83088.0
58049.8         64445.0
57499.0         57189.0
```

Practical 3:

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [2]:
```
# Load the dataset
data = pd.read_csv('Salary_Data.csv')
```

In [3]:
```
# Split the data into features (X) and labels (y)
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
```

In [4]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [5]:
```
# Create and train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

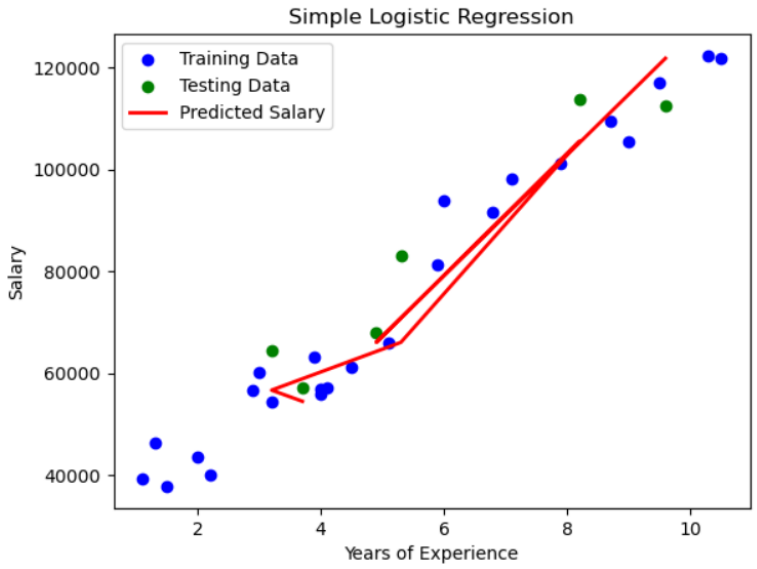
Out[5]:
```
LogisticRegression()
```

In [6]:
```
# Predict the labels for the test set
y_pred = model.predict(X_test)
```

In [7]:
```
# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.0
```

In [8]:
```
# Plot the logistic regression model
plt.scatter(X_train, y_train, color='blue', label='Training Data')
plt.scatter(X_test, y_test, color='green', label='Testing Data')
plt.plot(X_test, model.predict(X_test), color='red', linewidth=2, label='Predicted Salary')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Simple Logistic Regression')
plt.legend()
plt.show()
```



Practical 4:

In [10]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

In [11]:
```
# Load the dataset
data = pd.read_csv('Salary_Data.csv')
```

In [12]:
```
# Split the dataset into features (X) and labels (y)
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
```

In [4]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [5]:
```
# Create an SVM classifier
classifier = SVC(kernel='linear')
```

In [6]:
```
# Train the classifier
classifier.fit(X_train, y_train)
```

Out[6]:

SVC

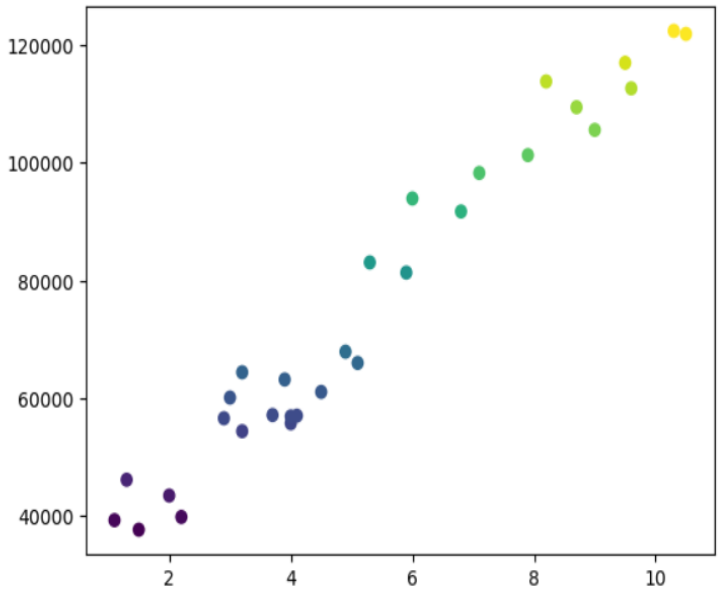
SVC(kernel='linear')

In [7]:
```
# Make predictions on the test set
y_pred = classifier.predict(X_test)
```

In [8]:
```
# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.0
```

In [16]:
```
# Plot the decision boundary
plt.scatter(X[:, 0], y, c=y, cmap='viridis')
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()
```



Practical 5:

In [11]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [12]:
```
# Load the dataset
data = pd.read_csv('Salary_Data.csv')
```

In [13]:
```
# Split the data into features and target variable
X = data.drop('Salary', axis=1)
y = data['Salary']
```

In [14]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [15]:
```
# Initialize the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [16]:
```
# Train the classifier
rf_classifier.fit(X_train, y_train)
```

Out[16]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [17]:
```
# Predict the target variable for the test set
y_pred = rf_classifier.predict(X_test)
```

In [18]:
```
# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.0
```

In [19]:
```
# Get feature importances
importances = rf_classifier.feature_importances_
features = X.columns
```

```
plt.bar(features, importances)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importances')
plt.xticks(rotation=45)
plt.show()
```