In [1]:
```
import pandas as pd
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

Applying Neural Network(ANN)

In [2]:
```
# Load the earthquake data using pandas DataFrame
df = pd.read_csv('database.csv')
```

In [3]:
```
# Split the data into training and testing datasets
X = df[['Latitude', 'Longitude']]
y = df['Magnitude']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

In [4]:
```
# Scale the data using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [5]:
```
# Create an Artificial Neural Network (ANN) model and fit the training data
regressor = MLPRegressor(hidden_layer_sizes=(50,50), max_iter=1000, alpha=0.0001,
                         solver='adam', random_state=42)
regressor.fit(X_train, y_train)
```

Out[5]:
```
MLPRegressor(hidden_layer_sizes=(50, 50), max_iter=1000, random_state=42)
```

In [6]:
```
# Predict the magnitude of earthquakes using the testing dataset
y_pred = regressor.predict(X_test)
```

In [7]:
```
# Evaluate the performance of the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 0.18523169876603948
```

Applying KNN

In [8]:
```
import pandas as pd
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

In [9]:
```
# Create a KNN model and fit the training data
regressor = KNeighborsRegressor(n_neighbors=3)
regressor.fit(X_train, y_train)
```

Out[9]:
```
KNeighborsRegressor(n_neighbors=3)
```

In [10]:
```
# Predict the magnitude of earthquakes using the testing dataset
y_pred = regressor.predict(X_test)
```

In [11]:
```
# Evaluate the performance of the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 0.23406175528507367
```

Applying Support Vector Machine

In [12]:
```
import pandas as pd
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

In [13]:
```
# Create a SVM model and fit the training data
regressor = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)
regressor.fit(X_train, y_train)
```

Out[13]:
```
SVR(C=100, gamma=0.1)
```

In [14]:
```
# Predict the magnitude of earthquakes using the testing dataset
y_pred = regressor.predict(X_test)
```

In [15]:
```
# Evaluate the performance of the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 0.19859993222142414
```

In [16]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
```

In [17]:
```
# Create a dictionary of models to test
models = {'Linear Regression': LinearRegression(),
          'Decision Tree': DecisionTreeRegressor(),
          'Random Forest': RandomForestRegressor(n_estimators=100),
          'Support Vector Machine': SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1),
          'K-Nearest Neighbors': KNeighborsRegressor(n_neighbors=5)}
```

In [18]:
```
# Fit each model to the training data and predict the magnitude of earthquakes using the
mse_results = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse_results[model_name] = mean_squared_error(y_test, y_pred)
```

In [19]:
```
# Print the MSE of each model
for model_name, mse in mse_results.items():
    print(f"{model_name} Mean Squared Error: {mse}")
```

```
Linear Regression Mean Squared Error: 0.18475232430091942
Decision Tree Mean Squared Error: 0.3521333546871664
Random Forest Mean Squared Error: 0.21213322327999162
Support Vector Machine Mean Squared Error: 0.19859993222142414
K-Nearest Neighbors Mean Squared Error: 0.21143874268631221
```

Conclusion:- Here Linear Regression performed well, it show less Mean Squared Error.