```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder
```

```
df = pd.read_csv('/content/Customer Purchasing Behaviors.csv')
df.head()
```

	user_id	age	annual_income	purchase_amount	loyalty_score	region	purchase_frequency
0	1	25	45000	200	4.5	North	12
1	2	34	55000	350	7.0	South	18
2	3	45	65000	500	8.0	West	22
3	4	22	30000	150	3.0	East	10
4	5	29	47000	220	4.8	North	13

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.shape
```

```
(238, 7)
```

```
# Prepare the data
X = df[['age', 'annual_income', 'purchase_amount', 'region', 'purchase_frequency']]
y = df['loyalty_score']
```

```
# One-hot encode the 'region' column
encoder = OneHotEncoder(handle_unknown='ignore')
encoded_region = encoder.fit_transform(X[['region']]).toarray()
feature_names = encoder.get_feature_names_out(['region'])
encoded_region_df = pd.DataFrame(encoded_region, columns=feature_names)
X = X.drop('region', axis=1)
X = pd.concat([X, encoded_region_df], axis=1)
```

```
# 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)
```

```
# Scale the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
class RBFN:
    def __init__(self, hidden_nodes, sigma=1.0):
        self.hidden_nodes = hidden_nodes
        self.sigma = sigma
        self.centers = None
        self.weights = None

    def gaussian_rbf(self, x, center):
        return np.exp(-np.linalg.norm(x - center) ** 2 / (2 * (self.sigma ** 2)))

    def fit(self, X, y):
        self.centers = X[np.random.choice(X.shape[0], self.hidden_nodes, replace=False)]
        hidden_layer_outputs = np.array([[self.gaussian_rbf(x, c) for c in self.centers] for x in X])
        self.weights = np.linalg.pinv(hidden_layer_outputs).dot(y)

    def predict(self, X):
        hidden_layer_outputs = np.array([[self.gaussian_rbf(x, c) for c in self.centers] for x in X])
        predictions = hidden_layer_outputs.dot(self.weights)
        return predictions
```

```
# Create and train the RBFN
rbfn = RBFN(hidden_nodes=10, sigma=1.0)
rbfn.fit(X_train, y_train)
```

```
# Make predictions on the test set
y_pred = rbfn.predict(X_test)
```

```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 16.196584687069585
```