```
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
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.datasets import make_regression
```

```
df= pd.read_csv("/content/tvmarketing.csv")
```

```
df.head()
```

	TV	Sales
0	230.1	22.1
1	44.5	10.4
2	17.2	9.3
3	151.5	18.5
4	180.8	12.9

Next steps: Generate code with df View recommended plots

```
df.tail()
```

	TV	Sales
195	38.2	7.6
196	94.2	9.7
197	177.0	12.8
198	283.6	25.5
199	232.1	13.4

```
df.shape
```

```
(200, 2)
```

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
TV	200.0	147.0425	85.854236	0.7	74.375	149.75	218.825	296.4
Sales	200.0	14.0225	5.217457	1.6	10.375	12.90	17.400	27.0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   TV      200 non-null    float64
 1   Sales   200 non-null    float64
dtypes: float64(2)
memory usage: 3.2 KB
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.datasets import make_regression
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.ensemble import GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor
```

```
X, y = make_regression(n_samples=100, n_features=5, noise=0.1, random_state=42)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


regressors = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "Support Vector Regressor": SVR(),
    "K-Neighbors Regressor": KNeighborsRegressor(),
    "Gradient Boosting Regressor": GradientBoostingRegressor(),
    "Neural Network" : MLPRegressor(max_iter=1000)
}
```

```
import warnings


for name, model in regressors.items():
    model.fit(X_train, y_train)
    scores = cross_val_score(model, X, y, cv=50, scoring='neg_mean_squared_error')


    mse_scores = -scores
    mean_mse = np.mean(mse_scores)
    std_mse = np.std(mse_scores)
    def adjusted_r2_score(r2, n, p):
      return 1 - (1 - r2) * ((n - 1) / (n - p - 1))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimize
  warnings.warn(
```

```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

```
r2 = r2_score(y_test, y_pred)
```

```
n=X_test.shape[0]
n
```

```
20
```

```
p = X_test.shape[1]
p
```

```
5
```

```
results = []
```

```
for name,model in regressors.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    n = X_test.shape[0]
    p = X_test.shape[1]
    adj_r2 = adjusted_r2_score(r2, n, p)
    mse = mean_squared_error(y_test,y_pred)
    mae = mean_absolute_error(y_test,y_pred)
    # Append results as dictionaries to the list
    results.append({"Regressor": name, "R2 Score": r2, "Adjusted R2 Score": adj_r2,"mean_squared_error":mse,"mean_absolute_error":mae})
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer:
  warnings.warn(
```

```
for name, model in regressors.items():
    print(f"--- {name} ---")

    metrics = next((item for item in results if item["Regressor"] == name), None)
    if metrics:
        for metric_name, value in metrics.items():
            if metric_name != 'Regressor':
                print(f"{metric_name}: {value:.4f}")
    print()

best_model_name = max(results, key=lambda k: k['Adjusted R2 Score'])
print(f"Best Model based on Adjusted R-squared: {best_model_name}")
```

```
--- Linear Regression ---
R2 Score: 1.0000
Adjusted R2 Score: 1.0000
mean_squared_error: 0.0113
mean_absolute_error: 0.0844

--- Decision Tree ---
R2 Score: 0.5186
Adjusted R2 Score: 0.3467
mean_squared_error: 9667.8080
mean_absolute_error: 77.4563

--- Random Forest ---
R2 Score: 0.7787
Adjusted R2 Score: 0.6996
mean_squared_error: 4444.9632
mean_absolute_error: 44.4607

--- Support Vector Regressor ---
R2 Score: 0.0405
Adjusted R2 Score: -0.3022
mean_squared_error: 19269.5792
mean_absolute_error: 104.6839

--- K-Neighbors Regressor ---
R2 Score: 0.7146
Adjusted R2 Score: 0.6127
mean_squared_error: 5730.6648
mean_absolute_error: 49.6629

--- Gradient Boosting Regressor ---
R2 Score: 0.8464
Adjusted R2 Score: 0.7916
mean_squared_error: 3084.1109
mean_absolute_error: 41.4867

--- Neural Network ---
R2 Score: 0.9696
Adjusted R2 Score: 0.9587
mean_squared_error: 611.2466
mean_absolute_error: 18.4604

Best Model based on Adjusted R-squared: {'Regressor': 'Linear Regression', 'R2 Score': 0.9999994350808352, 'Adjusted R2 Score': 0.99999
```

```
from pandas import DataFrame
results_df = pd.DataFrame(results)
```

```
print(results_df)
```

```
                     Regressor  R2 Score  Adjusted R2 Score  \
0            Linear Regression  0.999999           0.999999   
1                Decision Tree  0.518587           0.346654   
2                Random Forest  0.778661           0.699612   
3     Support Vector Regressor  0.040463          -0.302229   
4        K-Neighbors Regressor  0.714639           0.612725   
5  Gradient Boosting Regressor  0.846425           0.791577   
6               Neural Network  0.969563           0.958692   

   mean_squared_error  mean_absolute_error  
0            0.011345             0.084376  
1         9667.807992            77.456259  
2         4444.963239            44.460679  
3        19269.579169           104.683863  
4         5730.664764            49.662876  
5         3084.110912            41.486681  
6          611.246629            18.460399  
```

```
def adjusted_r2_score(r2, n, p):
    return 1 - (1 - r2) * ((n - 1) / (n - p - 1))
```

```
print(results_df)
```

```
                     Regressor  R2 Score  Adjusted R2 Score  \
0            Linear Regression  0.999999           0.999999   
1                Decision Tree  0.518587           0.346654   
2                Random Forest  0.778661           0.699612   
3     Support Vector Regressor  0.040463          -0.302229   
4        K-Neighbors Regressor  0.714639           0.612725   
5  Gradient Boosting Regressor  0.846425           0.791577   
```

```
6               Neural Network  0.969563           0.958692

   mean_squared_error  mean_absolute_error
0            0.011345             0.084376
1         9667.807992            77.456259
2         4444.963239            44.460679
3        19269.579169           104.683863
4         5730.664764            49.662876
5         3084.110912            41.486681
6          611.246629            18.460399
```

Start coding or generate with AI.