LAB 6: Support Vector Regression (SVR) – Insurance Dataset

2403A52006

MD.Mustafa

Import Required Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
```

Load Dataset

```
# Upload insurance.csv to Colab first
df = pd.read_csv("/content/insurance.csv")

df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1338,\n  \"fields\":
[\n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 14,\n        \"min\": 18,\n
\"max\": 64,\n        \"num_unique_values\": 47,\n        \"samples\":
[\n          21,\n          45,\n          36\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"male\",\n          \"female\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"bmi\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 6.098186911679017,\n
\"min\": 15.96,\n        \"max\": 53.13,\n
\"num_unique_values\": 548,\n        \"samples\": [\n          23.18,\
n          26.885\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"children\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 5,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"smoker\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"no\",\n          \"yes\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"region\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"southeast\",\n
\"northeast\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"charges\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 12110.011236693994,\n        \"min\": 1121.8739,\n
\"max\": 63770.42801,\n        \"num_unique_values\": 1337,\n
\"samples\": [\n          8688.85885,\n          5708.867\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


Encode Categorical Variables

```
df_encoded = pd.get_dummies(df, drop_first=True)

df_encoded.head()
```


```
{"summary":"{\n  \"name\": \"df_encoded\",\n  \"rows\": 1338,\n
\"fields\": [\n    {\n      \"column\": \"age\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
14,\n        \"min\": 18,\n        \"max\": 64,\n
\"num_unique_values\": 47,\n        \"samples\": [\n          21,\n
45,\n          36\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"bmi\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 6.098186911679017,\n        \"min\": 15.96,\n        \"max\":
53.13,\n        \"num_unique_values\": 548,\n        \"samples\": [\n
23.18,\n          26.885,\n          29.26\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"children\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 1,\n
\"min\": 0,\n        \"max\": 5,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          0,\n          1,\n          4\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"charges\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
12110.011236693994,\n        \"min\": 1121.8739,\n        \"max\":
63770.42801,\n        \"num_unique_values\": 1337,\n
\"samples\": [\n          8688.85885,\n          5708.867,\n
11436.73815\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sex_male\",\n      \"properties\": {\n        \"dtype\":
\"boolean\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          true,\n          false\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"smoker_yes\",\n
\"properties\": {\n        \"dtype\": \"boolean\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          false,\n
true\n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"region_northwest\",\n      \"properties\": {\n        \"dtype\":
\"boolean\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          true,\n          false\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"region_southeast\",\n
\"properties\": {\n        \"dtype\": \"boolean\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          true,\n
false\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"region_southwest\",\n      \"properties\": {\n        \"dtype\":
\"boolean\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          false,\n          true\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df_encoded"}
```


Select Features and Target

```
X = df_encoded.drop("charges", axis=1)   # Features
y = df_encoded["charges"]                # Target
```

Train-Test Split (80/20)

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

Apply Feature Scaling (Important for SVR)

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

SVR WITH RBF KERNEL

Train Model (RBF)

```
svr_rbf = SVR(kernel='rbf')
svr_rbf.fit(X_train_scaled, y_train)

y_pred_rbf = svr_rbf.predict(X_test_scaled)
```

Evaluate RBF Model

```
mae_rbf = mean_absolute_error(y_test, y_pred_rbf)
rmse_rbf = np.sqrt(mean_squared_error(y_test, y_pred_rbf))
r2_rbf = r2_score(y_test, y_pred_rbf)
```

```
print("SVR - RBF Kernel")
print("MAE:", mae_rbf)
print("RMSE:", rmse_rbf)
print("R2 Score:", r2_rbf)
```

```
SVR - RBF Kernel
MAE: 8612.408423351833
RMSE: 12889.096314656128
R2 Score: -0.07008155372454805
```

SVR WITH POLYNOMIAL KERNEL

Train Model (Polynomial)

```
svr_poly = SVR(kernel='poly', degree=2)
svr_poly.fit(X_train_scaled, y_train)

y_pred_poly = svr_poly.predict(X_test_scaled)
```

Evaluate Polynomial Model

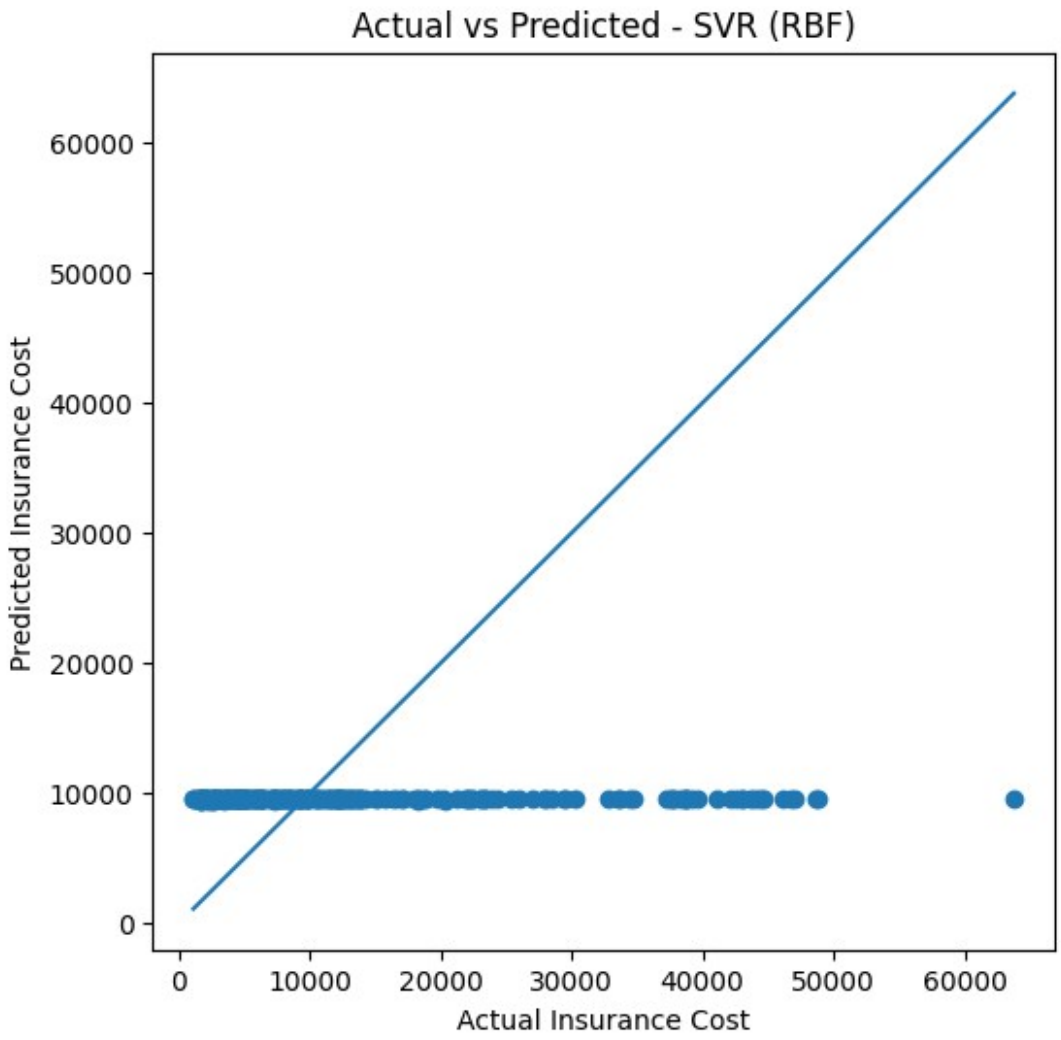
```
mae_poly = mean_absolute_error(y_test, y_pred_poly)
rmse_poly = np.sqrt(mean_squared_error(y_test, y_pred_poly))
r2_poly = r2_score(y_test, y_pred_poly)

print("SVR - Polynomial Kernel")
print("MAE:", mae_poly)
print("RMSE:", rmse_poly)
print("R2 Score:", r2_poly)
```

```
SVR - Polynomial Kernel
MAE: 8628.582335026595
RMSE: 12897.50266305726
R2 Score: -0.07147783653474571
```

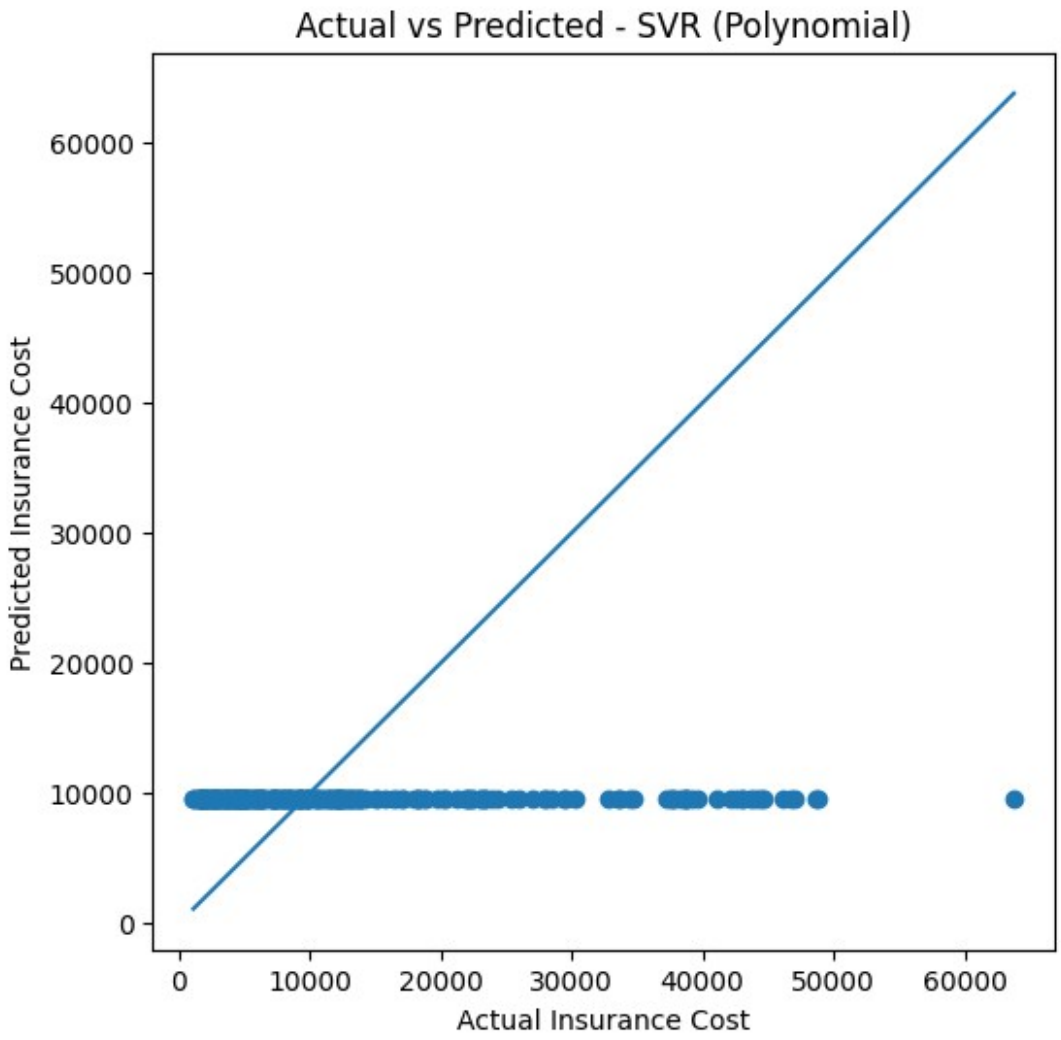
Actual vs Predicted (RBF)

```
plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred_rbf)
plt.xlabel("Actual Insurance Cost")
plt.ylabel("Predicted Insurance Cost")
plt.title("Actual vs Predicted - SVR (RBF)")
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()])
plt.show()
```



Actual vs Predicted (Polynomial)

```
plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred_poly)
plt.xlabel("Actual Insurance Cost")
plt.ylabel("Predicted Insurance Cost")
plt.title("Actual vs Predicted - SVR (Polynomial)")
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()])
plt.show()
```



Error Comparison Graph

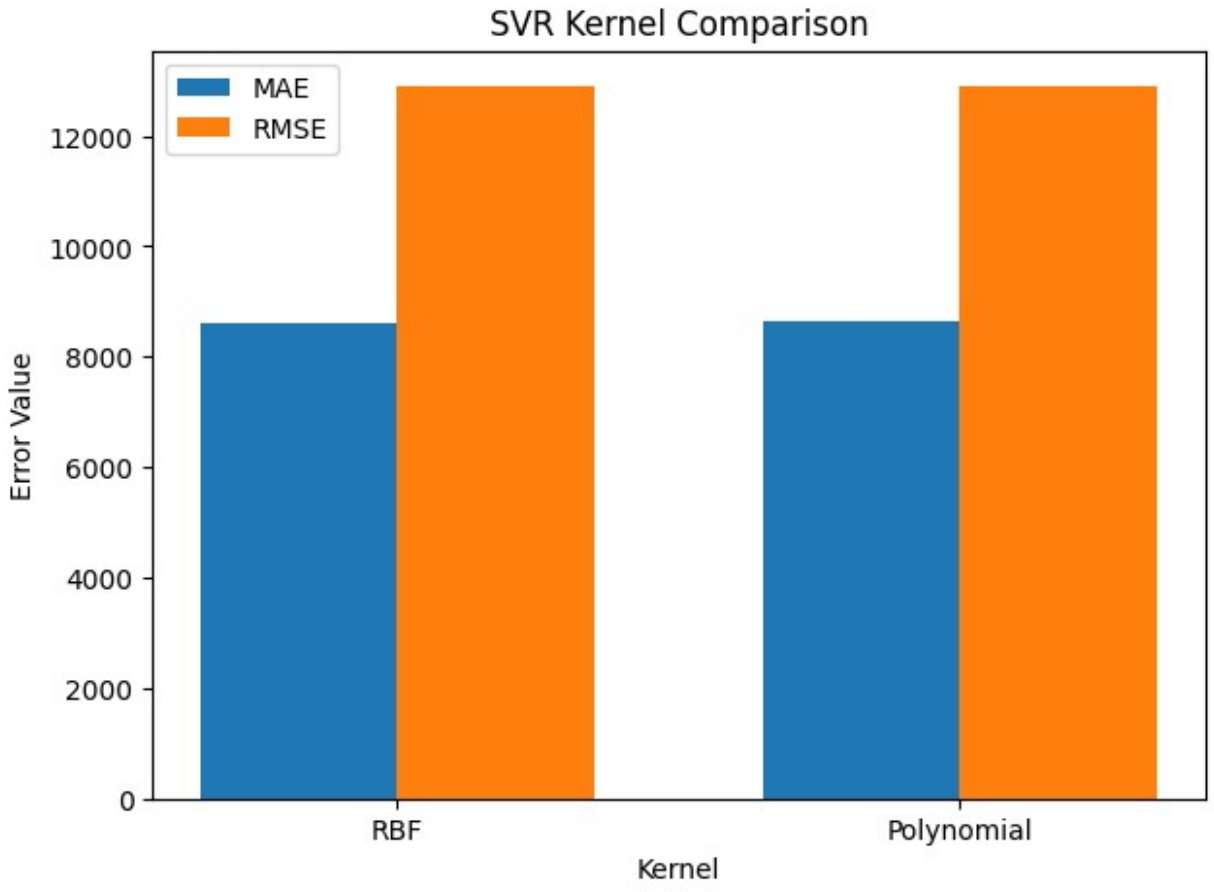
```
kernels = ['RBF', 'Polynomial']
mae_values = [mae_rbf, mae_poly]
rmse_values = [rmse_rbf, rmse_poly]

x = np.arange(len(kernels))
width = 0.35

plt.figure(figsize=(7,5))
plt.bar(x - width/2, mae_values, width, label='MAE')
plt.bar(x + width/2, rmse_values, width, label='RMSE')

plt.xticks(x, kernels)
plt.xlabel("Kernel")
plt.ylabel("Error Value")
plt.title("SVR Kernel Comparison")
```

```
plt.legend()
plt.show()
```



Final Comparison Table

```
results = pd.DataFrame({
    "Kernel": ["RBF", "Polynomial"],
    "MAE": [mae_rbf, mae_poly],
    "RMSE": [rmse_rbf, rmse_poly],
    "R2 Score": [r2_rbf, r2_poly]
})

results
```


```
{"summary":"{\n  \"name\": \"results\",\n  \"rows\": 2,\n  \"fields\":
[\n    {\n      \"column\": \"Kernel\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"Polynomial\",\n          \"RBF\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MAE\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 11.436682623535955,\n        \"min\": 8612.408423351833,\n
```

```
\"max\": 8628.582335026595,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          8628.582335026595,\n
8612.408423351833\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"RMSE\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 5.944185959456601,\n        \"min\": 12889.096314656128,\n
\"max\": 12897.50266305726,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          12897.50266305726,\n
12889.096314656128\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"R2
Score\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.0009873210435449727,\n        \"min\": -
0.07147783653474571,\n        \"max\": -0.07008155372454805,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          -
0.07147783653474571,\n          -0.07008155372454805\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"results"}
```


Conclusion

The SVR model with RBF kernel performs better as it produces lower MAE and RMSE

and higher R^2 score compared to the Polynomial kernel.

Hence, RBF kernel is best for predicting insurance cost.