```
import pandas as pd
df = pd.read_csv("insurance.csv")
print(df.head())
```

```
   age     sex     bmi  children smoker     region      charges
0   19  female  27.900         0    yes  southwest  16884.92400
1   18    male  33.770         1     no  southeast   1725.55230
2   28    male  33.000         3     no  southeast   4449.46200
3   33    male  22.705         0     no  northwest  21984.47061
4   32    male  28.880         0     no  northwest   3866.85520
```

```
categorical_cols = ['sex', 'smoker', 'region']
df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

X = df_encoded.drop('charges', axis=1)
y = df_encoded['charges']

print("First 5 rows of the encoded features (X):")
print(X.head())
print("\nFirst 5 rows of the target variable (y):")
print(y.head())
```

```
First 5 rows of the encoded features (X):
   age     bmi  children  sex_male  smoker_yes  region_northwest  \
0   19  27.900         0     False        True             False
1   18  33.770         1      True       False             False
2   28  33.000         3      True       False             False
3   33  22.705         0      True       False              True
4   32  28.880         0      True       False              True

   region_southeast  region_southwest
0             False              True
1              True             False
2              True             False
3             False             False
4             False             False

First 5 rows of the target variable (y):
0    16884.92400
1     1725.55230
2     4449.46200
3    21984.47061
4     3866.85520
Name: charges, dtype: float64
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

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

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (1070, 8)
Shape of X_test: (268, 8)
Shape of y_train: (1070,)
Shape of y_test: (268,)
```

```
from sklearn.svm import SVR

# Initialize SVR models with RBF and Polynomial kernels
svr_rbf = SVR(kernel='rbf')
svr_poly = SVR(kernel='poly')

# Train the svr_rbf model
svr_rbf.fit(X_train, y_train)
print("SVR RBF model trained successfully.")

# Train the svr_poly model
```

```
svr_poly.fit(X_train, y_train)
print("SVR Polynomial model trained successfully.")
```

```
SVR RBF model trained successfully.
SVR Polynomial model trained successfully.
```

```
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt

# --- Evaluate RBF SVR Model ---
# Make predictions
y_pred_rbf = svr_rbf.predict(X_test)

# Calculate metrics for 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("RBF SVR Model Performance:")
print(f"  Mean Absolute Error (MAE): {mae_rbf:.2f}")
print(f"  Root Mean Squared Error (RMSE): {rmse_rbf:.2f}")
print(f"  R-squared (R2) Score: {r2_rbf:.2f}")
print("\n")

# --- Evaluate Polynomial SVR Model ---
# Make predictions
y_pred_poly = svr_poly.predict(X_test)

# Calculate metrics for 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("Polynomial SVR Model Performance:")
print(f"  Mean Absolute Error (MAE): {mae_poly:.2f}")
print(f"  Root Mean Squared Error (RMSE): {rmse_poly:.2f}")
print(f"  R-squared (R2) Score: {r2_poly:.2f}")
print("\n")

# --- Visualize RBF Model Performance ---
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_rbf, alpha=0.6, color='blue', label='Predicted vs. Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2, label='Perfect Prediction')
plt.xlabel('Actual Charges')
plt.ylabel('Predicted Charges (RBF)')
plt.title('RBF SVR: Actual vs. Predicted Charges')
plt.legend()
plt.grid(True)
plt.show()

# --- Visualize Polynomial Model Performance ---
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_poly, alpha=0.6, color='green', label='Predicted vs. Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2, label='Perfect Prediction')
plt.xlabel('Actual Charges')
plt.ylabel('Predicted Charges (Polynomial)')
plt.title('Polynomial SVR: Actual vs. Predicted Charges')
plt.legend()
plt.grid(True)
plt.show()
```

```
RBF SVR Model Performance:
  Mean Absolute Error (MAE): 8612.41
  Root Mean Squared Error (RMSE): 12889.10
  R-squared (R2) Score: -0.07


Polynomial SVR Model Performance:
  Mean Absolute Error (MAE): 8607.80
  Root Mean Squared Error (RMSE): 12872.96
  R-squared (R2) Score: -0.07
```