

July 14, 2025

```
[4]: # Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.pipeline import Pipeline

# Load the dataset
print("Loading dataset...")
data = pd.read_csv('sp500_dataset.csv', index_col='Index', parse_dates=True)

# Data Exploration
print("\nDataset Overview:")
print(f"Shape: {data.shape}")
print(f"Columns: {list(data.columns)}")
print("\nMissing values per column:")
print(data.isna().sum().sort_values(ascending=False).head(10))

# Data Preprocessing
print("\nPreprocessing data...")
target_stock = 'AAPL' # Let's predict Apple's stock price
target = data['Company']

# Use other stocks' prices as features
features = data.drop(columns=[target_stock])

# Handle missing values - forward fill then backward fill
features = features.fillna(method='ffill').fillna(method='bfill')
target = target.fillna(method='ffill').fillna(method='bfill')

# Verify no missing values remain
print("\nMissing values after processing:")
print(f"Features: {features.isna().sum().sum()}")
print(f"Target: {target.isna().sum().sum()}")
```

```

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    features, target, test_size=0.2, shuffle=False, random_state=42)

print(f"\nTraining set size: {len(X_train)}")
print(f"Test set size: {len(X_test)}")

# Create SVM pipeline with standardization
print("\nBuilding SVM model...")
svm_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('svr', SVR(kernel='rbf', C=1.0, epsilon=0.1))
])

# Train the model
svm_pipeline.fit(X_train, y_train)

# Make predictions
y_train_pred = svm_pipeline.predict(X_train)
y_test_pred = svm_pipeline.predict(X_test)

# Evaluate model performance
train_mse = mean_squared_error(y_train, y_train_pred)
test_mse = mean_squared_error(y_test, y_test_pred)
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)

print("\nModel Performance Metrics:")
print(f"Training MSE: {train_mse:.2f}")
print(f"Test MSE: {test_mse:.2f}")
print(f"Training R2: {train_r2:.2f}")
print(f"Test R2: {test_r2:.2f}")

# Visualize predictions vs actual values
plt.figure(figsize=(14, 7))
plt.plot(y_test.index, y_test, label='Actual Prices', color='blue', linewidth=2)
plt.plot(y_test.index, y_test_pred, label='Predicted Prices', color='red',
        linestyle='--')
plt.title(f'{target_stock} Stock Price Prediction using SVM', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Price', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

```

```

# Feature importance (coefficients not directly available for SVR with RBF
↪kernel)
# For linear kernel, we could use:
# if svm_pipeline.named_steps['svr'].kernel == 'linear':
#     coef = svm_pipeline.named_steps['svr'].coef_
#     feature_importance = pd.Series(coef[0], index=features.columns)
#     feature_importance.sort_values().plot(kind='barh')
#     plt.title('Feature Importance (Linear Kernel)')
#     plt.show()

print("\nModel training succeed!")

```

Loading dataset...

Dataset Overview:

Shape: (499, 6)

Columns: ['Company', 'sector', 'annual_return_log', 'Std', 'Skewness', 'Kurtosis']

Missing values per column:

Company	0
sector	0
annual_return_log	0
Std	0
Skewness	0
Kurtosis	0
dtype:	int64

Preprocessing data...

C:\Users\WIN\AppData\Local\Temp\ipykernel_19968\1612278820.py:13: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.

```
data = pd.read_csv('sp500_dataset.csv', index_col='Index', parse_dates=True)
```

