Day53_Support_Vector_Machine_(SVM)_Classifier

July 28, 2025

Support Vector Machine (SVM) Classifier

Understanding SVM

- Suppose we have **two classes**, like `A` and `B`.
- **SVM** tries to find a line called the **hyperplane** that best separates these two classes.
- The points closest to the decision boundary are called **support vectors**.
- The distance between these support lines is called **marginal distance**.
- SVM tries to **maximize the margin** — which means better generalization and fewer errors.

Linear vs Non-linear SVM

- **Linear SVM**: Can separate classes using a straight line.
- **Non-linear SVM**: When data is not linearly separable, it uses a **kernel function** to project data into higher dimensions.
- This way, SVM converts a **non-linear problem** into a **linear one** in higher-dimensional space.

Examples:

- 1D → 2D
- 2D → 3D

Important SVM Concepts

- **Kernel function** helps move data to higher dimensions to make it separable.
- **Maximum margin = better generalization** (small errors can be adjusted, leading to more accuracy).
- Common kernels: `linear`, `rbf`, `poly`.

Now, let's implement an SVM Classifier in Python using scikit-learn!

1 Import Libraries

[2]:
```
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
```

```python
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
```

2 Load and Prepare Data

[3]:
```python
# Step 2: Load Dataset
dataset = pd.read_csv(r"C:\Users\Lenovo\Downloads\logit classification.csv")  #␣
↪adjust path if needed
dataset.head()
```

[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

3 Feature Selection & Train-Test Split

[4]:
```python
# Step 3: Select Features and Target
X = dataset[["Age", "EstimatedSalary"]].values
y = dataset["Purchased"].values
```

[5]:
```python
# Split into train-test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
↪random_state=0)
```

4 Feature Scaling

[6]:
```python
# Step 4: Scale Features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

5 Train SVM Classifier

[7]:
```python
# Step 5: Train the SVM model
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale')  # Try 'linear' or 'poly'␣
↪as well
svm_model.fit(X_train_scaled, y_train)
```

[7]: SVC()

6 Model Evaluation

[8]:
```python
# Step 6: Make Predictions and Evaluate
y_pred_svm = svm_model.predict(X_test_scaled)

print("SVM Classifier Results")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
print("Classification Report:\n", classification_report(y_test, y_pred_svm))
```

```
SVM Classifier Results
Accuracy: 0.93
Confusion Matrix:
 [[64  4]
 [ 3 29]]
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.94      0.95        68
           1       0.88      0.91      0.89        32

    accuracy                           0.93       100
   macro avg       0.92      0.92      0.92       100
weighted avg       0.93      0.93      0.93       100
```

7 Future Prediction

Once the model is trained, you can pass new user data (like Age and Estimated Salary) for prediction:

Example:

```python
new_data = [[30, 87000]]
scaled_data = scaler.transform(new_data)
svm_model.predict(scaled_data)
```

8 Predicting from Future Data CSV

[15]:
```python
# Step 7: Load Future Data for Prediction

import pandas as pd

future_data = pd.DataFrame({
    "User ID": [1674381, 1674382, 1674383, 1674384, 1674385, 1674386, 1674387,
                1674388, 1674389, 1674390, 1674391, 1674392, 1674393, 1674394],
    "Gender": ["Male", "Female", "Male", "Female", "Female", "Male", "Male",
               "Female", "Male", "Female", "Female", "Male", "Male", "Female"],
```

```
    "Age": [29, 14, 28, 58, 80, 90, 100, 45, 37, 48, 59, 60, 61, 62],
    "EstimatedSalary": [39000, 34500, 40000, 56490, 59000, 41000, 23000, 20000,
                        33000, 23000, 64000, 33000, 23000, 45000]
})



# or


# future_data = pd.read_csv(r"C:
 ↪\Users\Lenovo\OneDrive\Desktop\future_prediction.csv")

# Assuming your future file has columns: "Age", "EstimatedSalary"
print(future_data)
```

```
    User ID  Gender  Age  EstimatedSalary
0   1674381    Male   29            39000
1   1674382  Female   14            34500
2   1674383    Male   28            40000
3   1674384  Female   58            56490
4   1674385  Female   80            59000
5   1674386    Male   90            41000
6   1674387    Male  100            23000
7   1674388  Female   45            20000
8   1674389    Male   37            33000
9   1674390  Female   48            23000
10  1674391  Female   59            64000
11  1674392    Male   60            33000
12  1674393    Male   61            23000
13  1674394  Female   62            45000
```

9 Scale the future data using the same scaler used during training

[16]:
```
future_scaled = scaler.transform(future_data[["Age", "EstimatedSalary"]])
```

10 Predict using trained SVM model

[17]:
```
future_predictions = svm_model.predict(future_scaled)
```

11 Append predictions to the DataFrame

[18]:
```
future_data["Predicted_Purchase"] = future_predictions

print("\n Future Predictions:\n")
print(future_data)
```

```
 Future Predictions:

    User ID  Gender  Age  EstimatedSalary  Predicted_Purchase
0   1674381    Male   29            39000                   0
1   1674382  Female   14            34500                   0
2   1674383    Male   28            40000                   0
3   1674384  Female   58            56490                   1
4   1674385  Female   80            59000                   1
5   1674386    Male   90            41000                   1
6   1674387    Male  100            23000                   1
7   1674388  Female   45            20000                   1
8   1674389    Male   37            33000                   0
9   1674390  Female   48            23000                   1
10  1674391  Female   59            64000                   1
11  1674392    Male   60            33000                   1
12  1674393    Male   61            23000                   1
13  1674394  Female   62            45000                   1
```

Future Prediction Summary (SVM Classifier)

- **Goal:** Predict if a user will purchase based on **Age** and **EstimatedSalary**.
- **Output:** `Predicted_Purchase = 1` (Will buy), `0` (Will not buy)

Key Insights:

- **Older users (Age 58+)** are mostly predicted as buyers, even with lower salaries.
- **Younger users (<30)** are predicted as non-buyers, especially with average income.
- Model prioritizes **Age** more than **Salary** when making predictions.

Example:

Age	Salary	Prediction
29	39000	No
58	56490	Yes
90	41000	Yes

This helps companies **target real buyers** and reduce **marketing costs**.

12 Real-Time Use Case Explanation

How Does SVM Work in Real-Time Product Predictions?

Imagine a company like **Amazon** or **Flipkart** wants to predict whether a user will buy a product or not.
Here's how this real-time prediction happens using models like SVM:

Step-by-Step Real-Time Flow

1. **User visits the website**
   The system captures user features like:
   - Age
   - Estimated salary
   - Location, device, or browsing history
2. **Features are passed to a trained model**
   The features are **scaled** using the same scaler used during model training.
3. **Model gives prediction**
   The model outputs:
   - `1` (User is likely to purchase)
   - `0` (User is not likely to purchase)
4. **Personalization kicks in**
   - If the user is likely to buy → show **targeted offers, add-to-cart prompts**, urgency messages like "Only 1 left!"
   - If unlikely → offer **discounts, alternative recommendations**, etc.

Example

A user with:

Age	Estimated Salary
30	87,000

The system passes this to the model:

```
model.predict([[30, 87000]])
```

If **result = 1**, the product page shows:

"Special deal just for you!"

Business Impact

- Increases conversion rate
- Improves customer experience
- Reduces ad wastage by targeting the right users

13 Final Reflection

In this notebook, I explored how **Support Vector Machines (SVM)** can be used not just as a mathematical model, but as a **real-world decision-making tool**.

From training on historical data to making predictions on unseen users, this project shows how machine learning can:

- Improve business efficiency
- Personalize user experiences
- Reduce marketing costs
- Increase product conversions

This is just one use case. The same logic applies to **finance, healthcare, e-commerce, and more**.

This project boosted my understanding of:

- Classification models
- Real-time prediction systems
- Data preprocessing (scaling, splitting)
- Model evaluation and deployment thinking

"A model is only as good as the data and questions we ask of it."