SAVITRIBAI PHULE PUNE UNIVERSITY

A PROJECT REPORT ON

# PREDICTING PURCHASE PATTERNS USING SVM CLASSIFIER

BY

Yash Sonar 41381

Keshav Soni 41382

Under The Guidance of

Prof. P. S. Vidap

Class: BE 3



PUNE INSTITUTE OF COMPUTER TECHNOLOGY

Department of Computer Engineering Academic Year: 2019-20

**Abstract:**

The main aim of our project to study buying patterns of people using Support Vector Machines for Classification.

The dataset we used consists of the following features: age, salary and records of whether the person buys an item or not. We have used these features to predict whether a person of certain age and with given income will purchase an item or not. We used a linear kernel for the purpose of classification.

**Objective:**

The purposes of this project are:

1. To learn how to do data preprocessing.
2. To apply SVM classification algorithm.

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Content** | **Page No.** |
| 1 | Introduction | 4 |
| 2 | Data | 4 |
| 3 | Design/Implementation | 5 |
| 4 | Source Code | 5 |
| 5 | Output | 7 |
| 6 | Conclusion and Future Enhancement | 8 |

**Introduction:**

In classification, the idea is to predict the target class by analyzing the training dataset. This could be done by finding proper boundaries for each target class. In a general way of saying, use the training dataset to get better boundary conditions which could be used to determine each target class. Once the boundary conditions are determined, the next task is to predict the target class. The whole process is, therefore, known as classification.

In machine learning, support vector machines (SVMs, or support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

**Data:**

This is our input dataset containing 3 columns and 400 rows and ‘purchased’ is the dependent variable.

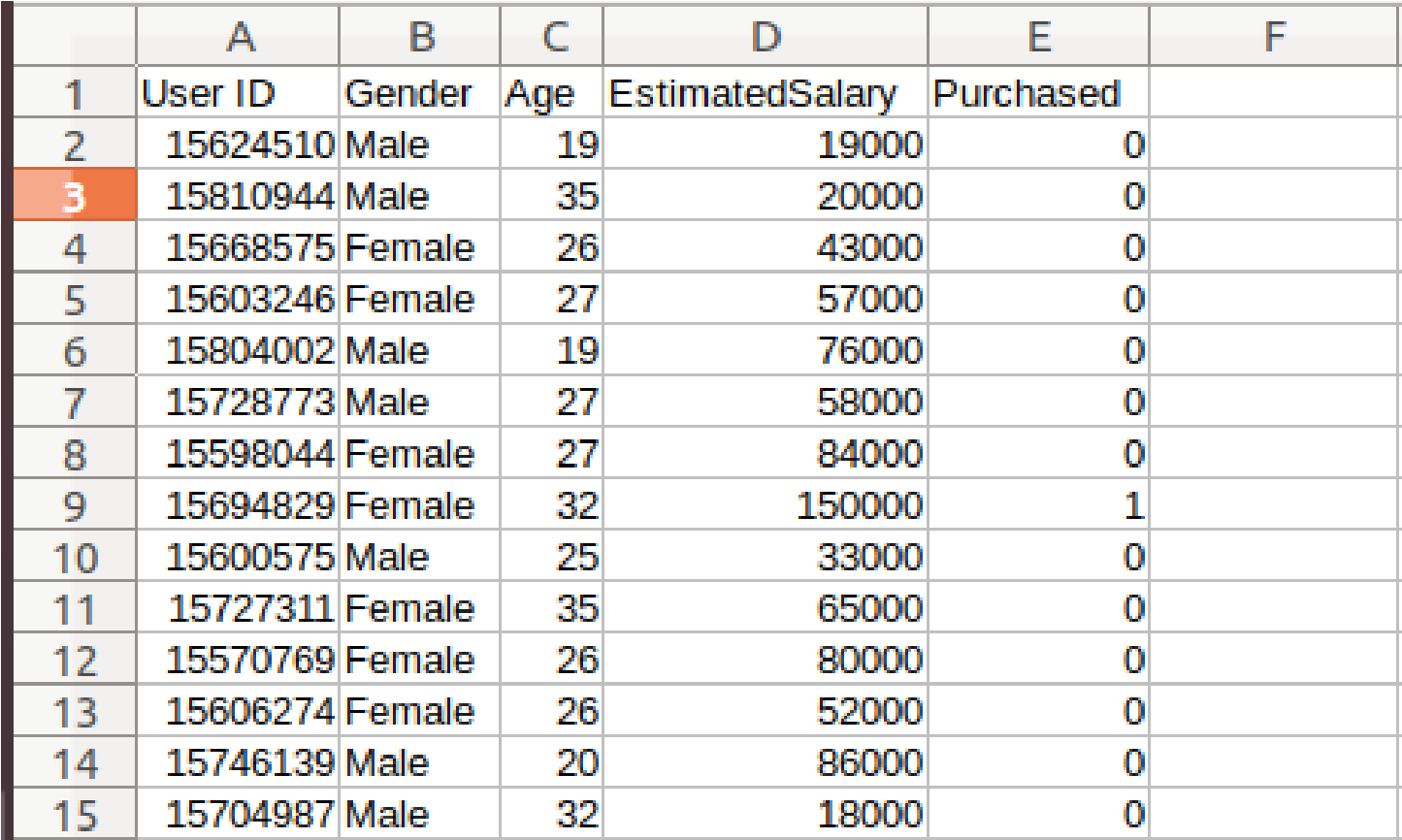


Fig. Dataset

**Design/Implementation:**

We first clean the data and than divide the dataset into training and testing sets. Then, we train the model and apply the classification model.

**Source Code:**

#Support Vector Machine (SVM) import numpy as np import matplotlib.pyplot as plt import pandas as pd from sklearn.model\_selection import cross\_validate from sklearn.model\_selection import train\_test\_split

#Import the dataset

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, [2, 3]].values y = dataset.iloc[:, 4].values

# Split the dataset into the Training set and Test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

#Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train) X\_test = sc.transform(X\_test)

#Fitting SVM to the Training set from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random\_state = 0) classifier.fit(X\_train, y\_train)

#Predicting the Test set results y\_pred = classifier.predict(X\_test)

#Making the Confusion Matrix from sklearn.metrics import confusion\_matrix cm = confusion\_matrix(y\_test, y\_pred)

#View the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01)) plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max()) plt.ylim(X2.min(), X2.max()) for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('SVM (Training set)') plt.xlabel('Age') plt.ylabel('Estimated Salary') plt.legend() plt.show()

#Visualise the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step

= 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01)) plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max()) plt.ylim(X2.min(), X2.max()) for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j) plt.title('SVM (Test set)') plt.xlabel('Age') plt.ylabel('Estimated Salary') plt.legend() plt.show()

**Output:**

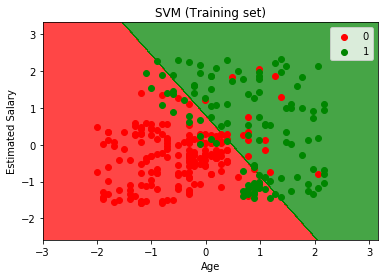


Fig: Training set

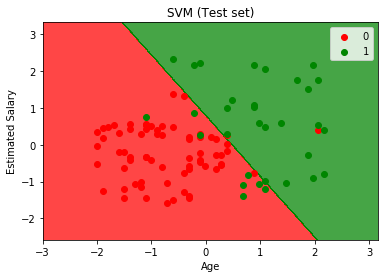


Fig: Testing set output

**Conclusion and Future Enhancement:**

To conclude, we have successfully implemented the project ‘Predicting Purchase Patterns using SVM classifier’.

For fututre work, more such classification algorithms can be studied and their accuracies can be compared with the this method.