

CIS 660 DATA MINING

Lab 3: Designing and Building a Prediction Model with a Classifier and Feature Selection with PCA tools (Extra Credit)

Fathima Syeda

CSU ID-2790024

Part 1: Feature Selection, Cleaning, and Preprocessing to Construct an Input from Data

Source:

The given data VTargetCustomerMail data consists of data about customers who had already purchased bikes in the past. This data consists of 18000 objects and a total of 32 features.

The screenshot shows an Excel spreadsheet titled 'VTargetCustomerMail - Excel'. The data is organized into columns labeled A through W. The first few columns contain customer identifiers and demographic information, while the remaining columns contain detailed educational and professional background data. The table is sorted by the 'CustomerKey' column.

| CustomerKey | GeographyKey | CustomerTitle | FirstName | MiddleName | LastName | NameStyle | BirthDate | MaritalStatus | Suffix | Gender | EmailAddress | YearlyIncome | TotalChildren | NumberChildren | EnglishEducation | SpanishEducation | FrenchEducation | EnglishOccupation | SpanishOccupation | FrenchOccupation | HouseOwner | NumberBikes |
|-------------|--------------|---------------|-----------|------------|-----------|-----------|-----------|---------------|----------|--------|--------------|--------------|---------------|----------------|------------------|------------------|-----------------|-------------------|-------------------|------------------|------------|-------------|
| 1 | 11000 | 26 | AW0001 | NULL | Jon | V | Yang | 0 | 4/8/1966 | M | jon24@aol | 90000 | 2 | 0 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 1 | 1 |
| 2 | 11001 | 37 | AW0001 | NULL | Eugene | L | Huang | 0 | ##### | S | NULL | 60000 | 3 | 3 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 0 | 0 |
| 3 | 11002 | 31 | AW0001 | NULL | Ruben | NULL | Torres | 0 | ##### | S | NULL | 60000 | 3 | 3 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 1 | 1 |
| 4 | 11003 | 11 | AW0001 | NULL | Christy | NULL | Zhu | 0 | ##### | S | NULL | 70000 | 0 | 0 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 0 | 0 |
| 5 | 11004 | 19 | AW0001 | NULL | Elizabeth | NULL | Johnson | 0 | 8/8/1968 | S | NULL | 80000 | 5 | 5 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 1 | 1 |
| 6 | 11005 | 22 | AW0001 | NULL | Julio | NULL | Rule | 0 | 5/5/1965 | S | NULL | 70000 | 0 | 0 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 1 | 1 |
| 7 | 11006 | 8 | AW0001 | NULL | Janet | G | Alvarez | 0 | ##### | S | NULL | 70000 | 0 | 0 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 1 | 1 |
| 8 | 11007 | 40 | AW0001 | NULL | Marco | NULL | Mehta | 0 | 5/9/1964 | M | NULL | 60000 | 3 | 3 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 1 | 1 |
| 9 | 11008 | 32 | AW0001 | NULL | Rob | NULL | Verhoff | 0 | 7/7/1964 | S | NULL | 60000 | 4 | 4 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 1 | 1 |
| 10 | 11009 | 25 | AW0001 | NULL | Shannon | C | Carlson | 0 | 4/7/1964 | S | NULL | 70000 | 0 | 0 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 0 | 0 |
| 11 | 11010 | 22 | AW0001 | NULL | Jacquelyn | C | Suarez | 0 | 2/6/1964 | S | NULL | 70000 | 0 | 0 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 0 | 0 |
| 12 | 11011 | 22 | AW0001 | NULL | Curtis | NULL | Lu | 0 | ##### | M | NULL | 60000 | 4 | 4 | Bachelors | Licenciatur | Bac + 4 | Professione | Professione | Cadre | 1 | 1 |
| 13 | 11012 | 611 | AW0001 | NULL | Lauren | M | Walker | 0 | ##### | M | NULL | 100000 | 2 | 2 | Bachelors | Licenciatur | Bac + 4 | Managemen | Managemen | Direction | 1 | 1 |
| 14 | 11013 | 543 | AW0001 | NULL | Ian | M | Jenkins | 0 | 6/6/1968 | M | NULL | 100000 | 2 | 2 | Bachelors | Licenciatur | Bac + 4 | Managemen | Managemen | Direction | 1 | 1 |
| 15 | 11014 | 634 | AW0001 | NULL | Sydney | NULL | Bennett | 0 | 5/9/1968 | S | NULL | 100000 | 3 | 3 | Bachelors | Licenciatur | Bac + 4 | Managemen | Managemen | Direction | 0 | 0 |
| 16 | 11015 | 301 | AW0001 | NULL | Chloe | NULL | Young | 0 | ##### | S | NULL | 30000 | 0 | 0 | Partial Col | Estudios u | Baccalaur | Skilled Ma | Obrero en | Technicien | 0 | 0 |
| 17 | 11016 | 329 | AW0001 | NULL | Wyatt | L | Hill | 0 | ##### | M | NULL | 30000 | 0 | 0 | Partial Col | Estudios u | Baccalaur | Skilled Ma | Obrero en | Technicien | 1 | 1 |
| 18 | 11017 | 39 | AW0001 | NULL | Shannon | NULL | Wang | 0 | ##### | S | NULL | 20000 | 4 | 4 | High Scho | Educacion | Bac + 2 | Skilled Ma | Obrero en | Technicien | 0 | 0 |
| 19 | 11018 | 32 | AW0001 | NULL | Clarence | D | Rai | 0 | ##### | S | NULL | 30000 | 2 | 2 | Partial Col | Estudios u | Baccalaur | Clerical | Administra | Employ | 1 | 1 |
| 20 | 11019 | 52 | AW0001 | NULL | Luke | L | Laf | 0 | 3/7/1978 | S | NULL | 40000 | 0 | 0 | High Scho | Educacion | Bac + 2 | Skilled Ma | Obrero en | Technicien | 0 | 0 |
| 21 | 11020 | 53 | AW0001 | NULL | Jordan | C | King | 0 | ##### | S | NULL | 40000 | 0 | 0 | High Scho | Educacion | Bac + 2 | Skilled Ma | Obrero en | Technicien | 0 | 0 |
| 22 | 11021 | 536 | AW0001 | NULL | Destiny | NULL | Wilson | 0 | 9/3/1978 | S | NULL | 40000 | 0 | 0 | Partial Col | Estudios u | Baccalaur | Skilled Ma | Obrero en | Technicien | 0 | 0 |
| 23 | 11022 | 609 | AW0001 | NULL | Ethan | G | Zhang | 0 | ##### | M | NULL | 40000 | 0 | 0 | Partial Col | Estudios u | Baccalaur | Skilled Ma | Obrero en | Technicien | 1 | 1 |
| 24 | 11023 | 298 | AW0001 | NULL | Seth | M | Edwards | 0 | ##### | M | NULL | 40000 | 0 | 0 | Partial Col | Estudios u | Baccalaur | Skilled Ma | Obrero en | Technicien | 1 | 1 |
| 25 | 11024 | 311 | AW0001 | NULL | Russell | NULL | Xie | 0 | ##### | M | NULL | 40000 | 0 | 0 | Partial Col | Estudios u | Baccalaur | Skilled Ma | Obrero en | Technicien | 1 | 1 |
| 26 | 11025 | 24 | AW0001 | NULL | Alejandro | NULL | Beck | 0 | ##### | M | NULL | 10000 | 2 | 2 | 1 Partial | Hig Educacion | Niveau bac | Clerical | Administra | Employ | 1 | 1 |
| 27 | 11026 | 4 | AW0001 | NULL | Harold | NULL | Sai | 0 | 4/3/1946 | S | NULL | 30000 | 2 | 2 | Partial Col | Estudios u | Baccalaur | Clerical | Administra | Employ | 0 | 0 |
| 28 | 11027 | 40 | AW0001 | NULL | Jessie | R | Zhao | 0 | ##### | M | NULL | 30000 | 2 | 2 | Partial Col | Estudios u | Baccalaur | Clerical | Administra | Employ | 1 | 1 |
| 29 | 11028 | 17 | AW0001 | NULL | Ill | NULL | Benner | 0 | ##### | M | NULL | 30000 | 2 | 2 | Partial Col | Estudios u | Baccalaur | Clerical | Administra | Employ | 1 | 1 |

We must get a list of potential buyers from the given data. Of the given features we choose those features that are most relevant in finding the target buyers list.

We choose 12 features that are most relevant in predicting future bike buyers.

The following shows the VTargetBuyersList file with the chosen features.

The screenshot shows an Excel spreadsheet titled 'VTargetBuyersList - Excel'. The data is organized into columns labeled A through W. The first few columns contain customer identifiers and demographic information, while the remaining columns contain detailed educational and professional background data. The table is sorted by the 'CustomerKey' column.

| CustomerKey | GeographyKey | MaritalStatus | Gender | YearlyIncome | TotalChildren | NumberChildren | EnglishEducation | SpanishEducation | FrenchEducation | HouseOwner | NumberBikes | CommuteRegion | Age | BikeBuyer |
|-------------|--------------|---------------|--------|--------------|---------------|----------------|------------------|------------------|-----------------|------------|-------------|---------------|-----|-----------|
| 1 | 11000 | 26 | M | 90000 | 2 | 0 | Bachelors | Professione | 1 | 0 | 1-2 Miles | Pacific | 49 | 1 |
| 2 | 11001 | 37 | S | 60000 | 3 | 3 | Bachelors | Professione | 0 | 1 | 0-1 Miles | Pacific | 50 | 1 |
| 3 | 11002 | 31 | M | 60000 | 3 | 3 | Bachelors | Professione | 1 | 1 | 1-2 Miles | Pacific | 50 | 1 |
| 4 | 11003 | 11 | S | 70000 | 0 | 0 | Bachelors | Professione | 0 | 1 | 1-5 Miles | Pacific | 47 | 1 |
| 5 | 11004 | 19 | S | 80000 | 5 | 5 | Bachelors | Professione | 1 | 4 | 1-2 Miles | Pacific | 47 | 1 |
| 6 | 11005 | 22 | S | 70000 | 0 | 0 | Bachelors | Professione | 1 | 1 | 1-5 Miles | Pacific | 50 | 1 |
| 7 | 11006 | 8 | S | 70000 | 0 | 0 | Bachelors | Professione | 1 | 1 | 1-5 Miles | Pacific | 49 | 1 |
| 8 | 11007 | 40 | M | 60000 | 3 | 3 | Bachelors | Professione | 1 | 2 | 0-1 Miles | Pacific | 51 | 1 |
| 9 | 11008 | 32 | S | 60000 | 4 | 4 | Bachelors | Professione | 1 | 3 | 10+ Miles | Pacific | 51 | 1 |
| 10 | 11009 | 25 | S | 70000 | 0 | 0 | Bachelors | Professione | 0 | 1 | 1-5 Miles | Pacific | 51 | 1 |
| 11 | 11010 | 22 | S | 70000 | 0 | 0 | Bachelors | Professione | 0 | 1 | 1-5 Miles | Pacific | 51 | 1 |
| 12 | 11011 | 22 | M | 60000 | 4 | 4 | Bachelors | Professione | 1 | 4 | 10+ Miles | Pacific | 51 | 1 |
| 13 | 11012 | 611 | M | 100000 | 2 | 2 | Bachelors | Managemen | 1 | 2 | 1-2 Miles | North Ame | 47 | 0 |
| 14 | 11013 | 543 | M | 100000 | 2 | 2 | Bachelors | Managemen | 1 | 3 | 0-1 Miles | North Ame | 47 | 0 |
| 15 | 11014 | 634 | S | 100000 | 3 | 3 | Bachelors | Managemen | 0 | 3 | 1-2 Miles | North Ame | 47 | 0 |
| 16 | 11015 | 301 | S | 30000 | 0 | 0 | Partial Col | Skilled Ma | 0 | 1 | 1-5 Miles | North Ame | 36 | 1 |
| 17 | 11016 | 329 | M | 30000 | 0 | 0 | Partial Col | Skilled Ma | 1 | 1 | 1-5 Miles | North Ame | 36 | 1 |
| 18 | 11017 | 39 | S | 20000 | 4 | 4 | High Scho | Skilled Ma | 1 | 2 | 5-10 Miles | Pacific | 71 | 1 |
| 19 | 11018 | 32 | S | 30000 | 2 | 2 | Partial Col | Clerical | 1 | 2 | 5-10 Miles | Pacific | 70 | 1 |
| 20 | 11019 | 52 | S | 40000 | 0 | 0 | High Scho | Skilled Ma | 0 | 2 | 5-10 Miles | North Ame | 37 | 0 |
| 21 | 11020 | 53 | M | 40000 | 0 | 0 | High Scho | Skilled Ma | 0 | 2 | 1-2 Miles | North Ame | 36 | 1 |
| 22 | 11021 | 536 | S | 40000 | 0 | 0 | Partial Col | Skilled Ma | 0 | 1 | 1-2 Miles | North Ame | 37 | 1 |
| 23 | 11022 | 609 | M | 40000 | 0 | 0 | Partial Col | Skilled Ma | 1 | 1 | 1-5 Miles | North Ame | 36 | 1 |
| 24 | 11023 | 298 | M | 40000 | 0 | 0 | Partial Col | Skilled Ma | 1 | 1 | 1-2 Miles | North Ame | 36 | 0 |
| 25 | 11024 | 311 | M | 60000 | 0 | 0 | Partial Col | Skilled Ma | 1 | 2 | 5-10 Miles | North Ame | 36 | 0 |
| 26 | 11025 | 24 | M | 10000 | 2 | 2 | 1 Partial | Hig Clerical | 1 | 2 | 1-2 Miles | Pacific | 69 | 1 |
| 27 | 11026 | 4 | S | 30000 | 2 | 2 | Partial Col | Clerical | 0 | 2 | 1-2 Miles | Pacific | 69 | 1 |
| 28 | 11027 | 40 | M | 30000 | 2 | 2 | Partial Col | Clerical | 1 | 2 | 5-10 Miles | Pacific | 68 | 1 |
| 29 | 11028 | 17 | M | 30000 | 2 | 2 | Partial Col | Clerical | 1 | 2 | 1-2 Miles | Pacific | 69 | 1 |

File

Home

Insert

Page Layout

Formulas

Data

Review

View

Add-Ins

Tell me what you want to do...

Share

Clipboard

Cut

Copy

Format Painter

Font

B

I

U

Color

Background Color

Alignment

Merge & Center

Number

Conditional Formatting

Normal

Bad

Good

Neutral

Calculation

Check Cell

Insert

Delete Format

AutoSum

Fill

Clear

Sort & Find & Filter

Select

Cells

Editing

A1

MaritalStatus

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |
|----|---------------|--------|--------------|---------------|----------------|---------|------|------------|----------|------------|---------|------|-----------|-----------|-----------|-----------|---|---|---|---|---|---|---|
| 1 | MaritalStatus | Gender | YearlyIncome | TotalChildren | NumberChildren | English | Edu | EnglishOcc | HouseOwn | NumberCars | Commute | Age | Region_Eu | Region_Nr | Region_Pa | BikeBuyer | | | | | | | |
| 2 | 0 | 1 | 0.4 | 0 | 0 | 0.75 | 0.75 | 0 | 0.25 | 0 | 0.5 | 0.25 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 3 | 1 | 0 | 0.5 | 0.6 | 0.6 | 0.75 | 0.75 | 0 | 0.25 | 0 | 0.5 | 0 | 0.5 | 0 | 0 | 1 | 1 | | | | | | |
| 4 | 1 | 0 | 1 | 0.5 | 0.6 | 0.6 | 0.75 | 0.75 | 1 | 0.25 | 0.5 | 0.5 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 5 | 1 | 0 | 0.75 | 0 | 0 | 0.75 | 0.75 | 0 | 0.25 | 0.75 | 0.25 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 6 | 1 | 0 | 0.75 | 1 | 1 | 0.75 | 0.75 | 1 | 1 | 0.25 | 0.25 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 7 | 1 | 1 | 0.75 | 0 | 0 | 0.75 | 0.75 | 1 | 0.25 | 0.75 | 0.5 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 8 | 1 | 0 | 0.75 | 0 | 0 | 0.75 | 0.75 | 1 | 0.25 | 0.75 | 0.25 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 9 | 0 | 1 | 0.5 | 0.6 | 0.6 | 0.75 | 0.75 | 1 | 0.5 | 0 | 0.5 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 10 | 1 | 0 | 0.5 | 0.8 | 0.8 | 0.75 | 0.75 | 1 | 0.75 | 1 | 0.5 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 11 | 1 | 1 | 0.75 | 0 | 0 | 0.75 | 0.75 | 0 | 0.25 | 0.75 | 0.5 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 12 | 1 | 0 | 0.75 | 0 | 0 | 0.75 | 0.75 | 0 | 0.25 | 0.75 | 0.5 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 13 | 0 | 1 | 0.5 | 0.8 | 0.8 | 0.75 | 0.75 | 1 | 1 | 1 | 0.5 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 14 | 0 | 1 | 0.4 | 1 | 0 | 0.75 | 1 | 1 | 0.5 | 0.25 | 0.25 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 15 | 0 | 1 | 1 | 0.4 | 0 | 0.75 | 1 | 1 | 0.75 | 0 | 0.25 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 16 | 1 | 0 | 1 | 0.6 | 0 | 0.75 | 1 | 0 | 0.75 | 0.25 | 0.25 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 17 | 1 | 0 | 0 | 0 | 0 | 0.5 | 0.25 | 0 | 0.25 | 0.75 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 18 | 0 | 1 | 0 | 0 | 0 | 0.5 | 0.25 | 1 | 0.25 | 0.75 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 19 | 1 | 0 | 0 | 0.8 | 0 | 0.25 | 0.25 | 1 | 0.5 | 0.75 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 20 | 1 | 1 | 0 | 0.4 | 0.5 | 0.5 | 0.5 | 1 | 0.5 | 0.75 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 21 | 1 | 1 | 0.25 | 0 | 0 | 0.25 | 0.25 | 0 | 0.5 | 0.75 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 22 | 1 | 1 | 0.25 | 0 | 0 | 0.25 | 0.25 | 0 | 0.5 | 0.25 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 23 | 1 | 0 | 0.25 | 0 | 0 | 0.5 | 0.25 | 0 | 0.25 | 0.25 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 24 | 0 | 1 | 0.25 | 0 | 0 | 0.5 | 0.25 | 1 | 0.25 | 0.75 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 25 | 0 | 1 | 0.25 | 0 | 0 | 0.5 | 0.25 | 1 | 0.25 | 0.25 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 26 | 0 | 1 | 0.5 | 0 | 0 | 0.5 | 0.25 | 1 | 0.5 | 0.75 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 27 | 0 | 1 | 0 | 0.4 | 0.2 | 0 | 0.5 | 1 | 0.5 | 0.25 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 28 | 1 | 1 | 0 | 0.4 | 0 | 0.5 | 0.5 | 0 | 0.5 | 0.25 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 29 | 0 | 1 | 0 | 0.4 | 0 | 0.5 | 0.5 | 1 | 0.5 | 0.75 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| 30 | | | 0 | 0.4 | 0 | 0.5 | 0.5 | 1 | 0.5 | 0.25 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |

NormalizedData

Windows Taskbar

Taskbar

System Tray

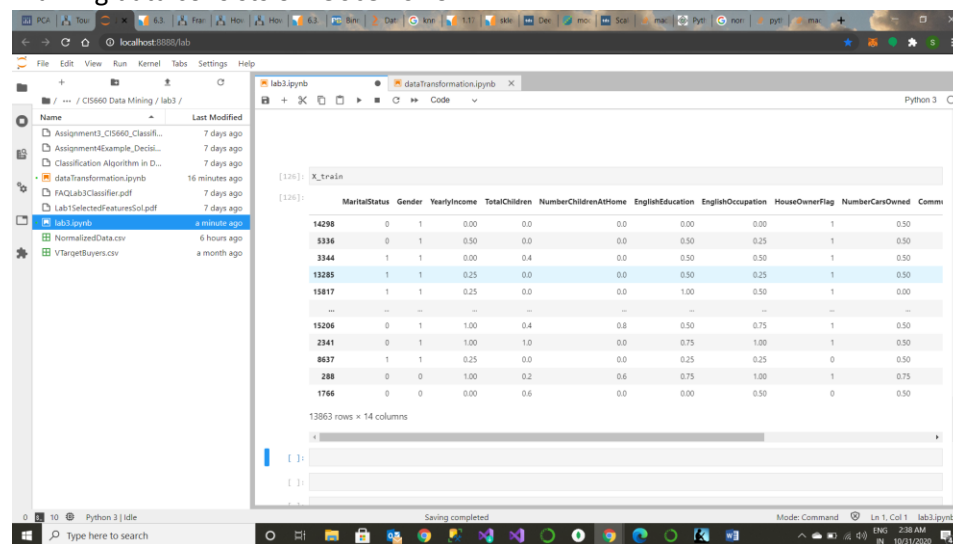
PART-2 1 Data Analytic Experiment with Two different Classifiers of Your Choice.

In this section we perform classification of the pre-processed data using two classifiers SVM and Neural Networks using various settings.

Before classifying the Transformed(pre-processed) data we remove the **BikeBuyer** column from the dataset as Labels.

The dataset is also split into train-test dataset using **sklearn.model_selection import train_test_split**, where the test set size is 25 % of the entire dataset and the rest is train set .

Training data consists of 13863 rows

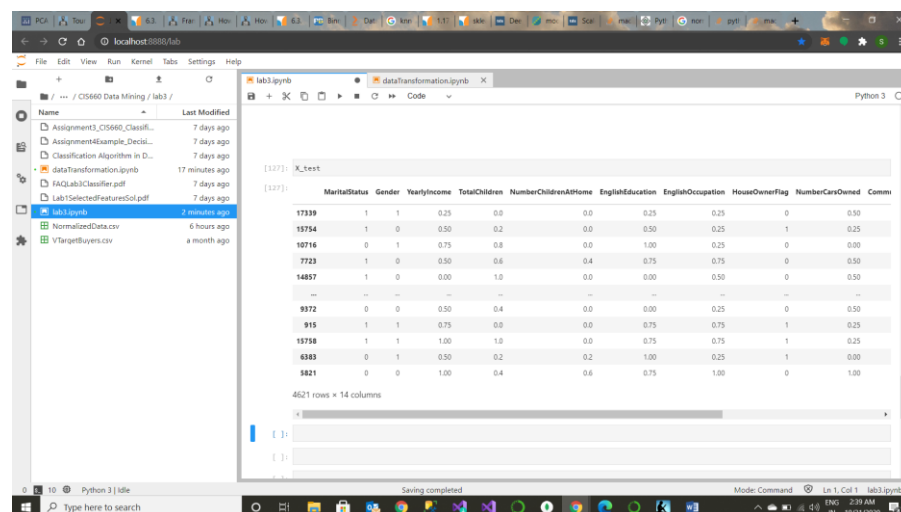


The screenshot shows a JupyterLab interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like 'dataTransformation.ipynb', 'NormalizedData.csv', and 'VTargetBuyers.csv'. The code editor shows a Jupyter notebook with a cell containing the following data:

| MaritalStatus | Gender | YearlyIncome | TotalChildren | NumberChildrenAtHome | EnglishEducation | EnglishOccupation | HouseOwnerFlag | NumberCarsOwned | Comm |
|---------------|--------|--------------|---------------|----------------------|------------------|-------------------|----------------|-----------------|------|
| 14298 | 0 | 1 | 0.00 | 0.0 | 0.0 | 0.00 | 0.00 | 1 | 0.50 |
| 5336 | 0 | 1 | 0.50 | 0.0 | 0.0 | 0.50 | 0.25 | 1 | 0.50 |
| 3344 | 1 | 1 | 0.00 | 0.4 | 0.0 | 0.50 | 0.50 | 1 | 0.50 |
| 13285 | 1 | 1 | 0.25 | 0.0 | 0.0 | 0.50 | 0.25 | 1 | 0.50 |
| 15817 | 1 | 1 | 0.25 | 0.0 | 0.0 | 0.50 | 1.00 | 1 | 0.00 |
| 15206 | 0 | 1 | 1.00 | 0.4 | 0.8 | 0.50 | 0.75 | 1 | 0.50 |
| 2341 | 0 | 1 | 1.00 | 1.0 | 0.0 | 0.75 | 1.00 | 1 | 0.50 |
| 8637 | 1 | 1 | 0.25 | 0.0 | 0.0 | 0.25 | 0.25 | 0 | 0.50 |
| 288 | 0 | 0 | 1.00 | 0.2 | 0.6 | 0.75 | 1.00 | 1 | 0.75 |
| 1766 | 0 | 0 | 0.00 | 0.6 | 0.0 | 0.00 | 0.50 | 0 | 0.50 |

13863 rows x 14 columns

Test data consists of 4621 rows



The screenshot shows a JupyterLab interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like 'dataTransformation.ipynb', 'NormalizedData.csv', and 'VTargetBuyers.csv'. The code editor shows a Jupyter notebook with a cell containing the following data:

| MaritalStatus | Gender | YearlyIncome | TotalChildren | NumberChildrenAtHome | EnglishEducation | EnglishOccupation | HouseOwnerFlag | NumberCarsOwned | Comm |
|---------------|--------|--------------|---------------|----------------------|------------------|-------------------|----------------|-----------------|------|
| 17339 | 1 | 1 | 0.25 | 0.0 | 0.0 | 0.25 | 0.25 | 0 | 0.50 |
| 15754 | 1 | 0 | 0.50 | 0.2 | 0.0 | 0.50 | 0.25 | 1 | 0.25 |
| 10716 | 0 | 1 | 0.75 | 0.8 | 0.0 | 1.00 | 0.25 | 0 | 0.00 |
| 7723 | 1 | 0 | 0.50 | 0.6 | 0.4 | 0.75 | 0.75 | 0 | 0.50 |
| 14857 | 1 | 0 | 0.00 | 1.0 | 0.0 | 0.00 | 0.50 | 0 | 0.50 |
| 9372 | 0 | 0 | 0.50 | 0.4 | 0.0 | 0.00 | 0.25 | 0 | 0.50 |
| 915 | 1 | 1 | 0.75 | 0.0 | 0.0 | 0.75 | 0.75 | 1 | 0.25 |
| 15758 | 1 | 1 | 1.00 | 1.0 | 0.0 | 0.75 | 0.75 | 1 | 0.25 |
| 6383 | 0 | 1 | 0.50 | 0.2 | 0.2 | 1.00 | 0.25 | 1 | 0.00 |
| 5821 | 0 | 0 | 1.00 | 0.4 | 0.6 | 0.75 | 1.00 | 0 | 1.00 |

4621 rows x 14 columns

SVM classifier :

The pre-processed data is fed into the SVM classifier on three times with various kernel settings. The kernel values used in the experiment are – linear, rbf and polynomial. The gamma value for SVM is set to 'scale' all throughout the classifications.

SVM classifier works by vectorization of each record. The following are the Accuracy and False positive and Miss Rate values when the data is classified using various Kernel settings of the SVM.

For SVM classified with kernel=linear

Accuracy:63.83899588833586

False positive rate:0.30848329048843187

Miss Rate:0.41582859641451686

For SVM classified with kernel=rbf

Accuracy:71.2832720190435

False positive rate:0.23221936589545844

Miss Rate:0.34324442501093133

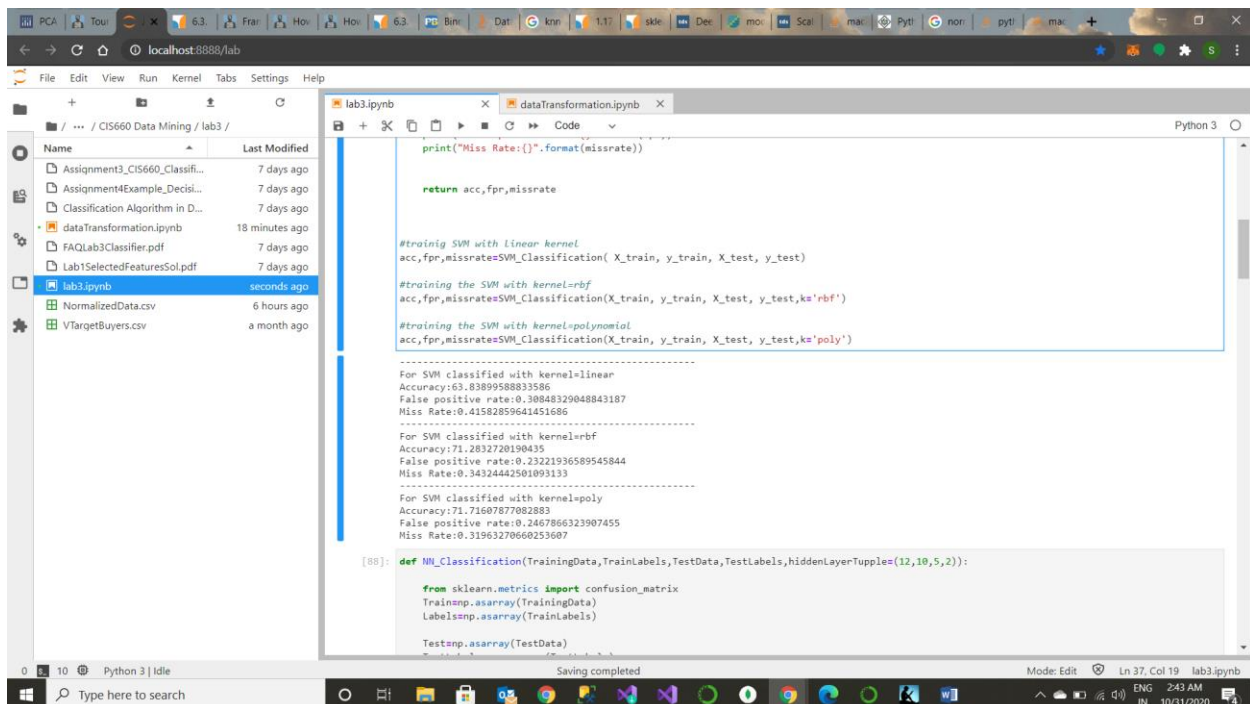
For SVM classified with kernel=poly

Accuracy:71.71607877082883

False positive rate:0.2467866323907455

Miss Rate:0.31963270660253607

The highest Accuracy is with SVM kernels= rbf and polynomial. This is because the



```
print("Miss Rate:{}".format(missrate))

return acc,fpr,missrate

#training SVM with linear kernel
acc,fpr,missrate=SVM_Classification( X_train, y_train, X_test, y_test)

#training the SVM with kernel=rbf
acc,fpr,missrate=SVM_Classification(X_train, y_train, X_test, y_test,ks='rbf')

#training the SVM with kernel=polynomial
acc,fpr,missrate=SVM_Classification(X_train, y_train, X_test, y_test,ks='poly')

For SVM classified with kernel=linear
Accuracy:63.83899588833586
False positive rate:0.30848329048843187
Miss Rate:0.41582859641451686
-----
For SVM classified with kernel=rbf
Accuracy:71.2832720190435
False positive rate:0.23221936589545844
Miss Rate:0.34324442501093133
-----
For SVM classified with kernel=poly
Accuracy:71.71607877082883
False positive rate:0.2467866323907455
Miss Rate:0.31963270660253607

[88]: def NN_Classification(TrainingData,TrainLabels,TestData,TestLabels,hiddenLayerTuple=(12,10,5,2)):

from sklearn.metrics import confusion_matrix
Train=np.asarray(TrainingData)
Labels=np.asarray(TrainLabels)

Test=np.asarray(TestData)
```

Neural Network classifier :

Three neural network topologies are designed with different hidden layers and different units in each layer. The neural network topologies are:

- NN with hidden layers=3 having (12, 10, 5) neurons
- NN classified with hidden layers=3 having (50, 25, 12) neurons
- NN classified with hidden layers=4 having (75, 25, 12, 5) neurons

For NN classified with hidden layers=3 having (12, 10, 5) neurons

Accuracy:71.9974031594893

False positive rate:0.23005487547488393

Miss Rate:0.33259325044404975

For NN classified with hidden layers=3 having (50, 25, 12) neurons

Accuracy:75.89266392555723

False positive rate:0.16040523427606584

Miss Rate:0.3259325044404973

For NN classified with hidden layers=4 having (75, 25, 12, 5) neurons

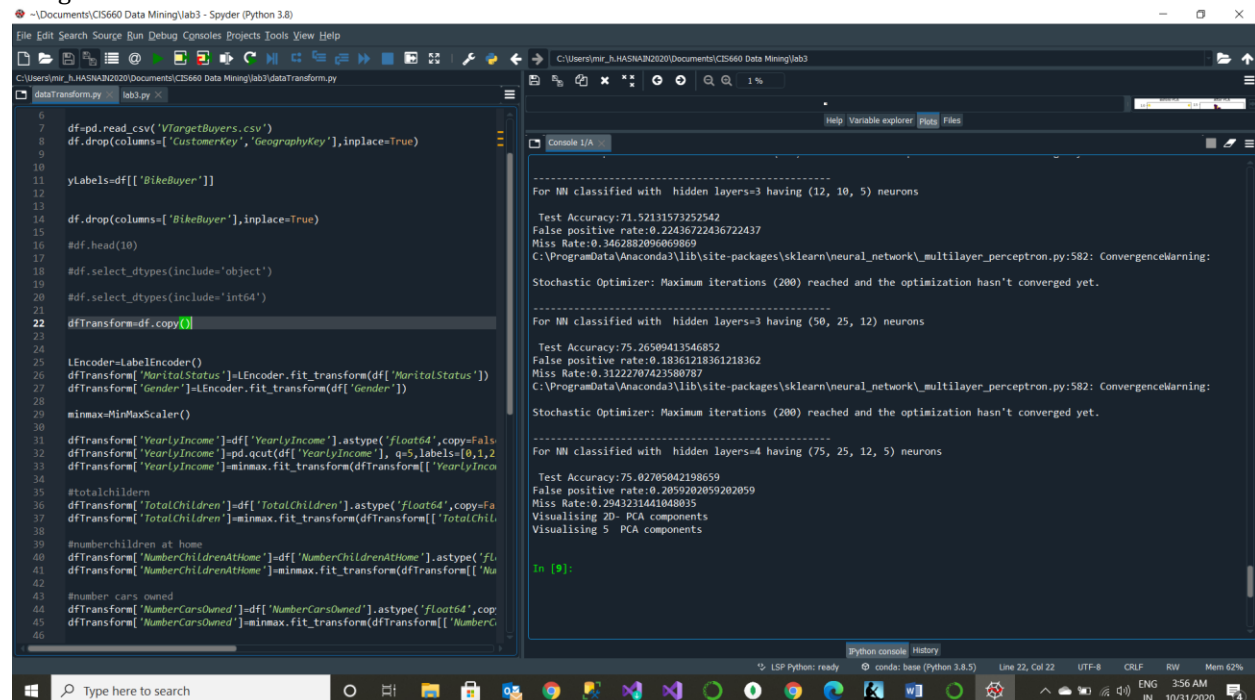
Accuracy:75.80610257520017

False positive rate:0.25707049387927394

Miss Rate:0.22602131438721138

The highest accuracy is with the Neural network topology having hidden layers=3 having (50, 25, 12) units, followed by a very close margin by the neural network having hidden layers=4 having (75, 25, 12, 5) units.

Thus we see that after a point the increase in number of layers doesn't impact the accuracy much and counter intuitively adding too many layers and units could increase the execution time but not increase the accuracy or might even reduce it.



```
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\mir_h\ASANA\2020\Documents\CS5660 Data Mining\lab3
dataTransform.py lab3.py
6 df=pd.read_csv('VTargetBuyers.csv')
7 df.drop(columns=['CustomerKey','GeographyKey'],inplace=True)
8
9
10
11 yLabels=df[['BikeBuyer']]
12
13
14 df.drop(columns=['BikeBuyer'],inplace=True)
15
16 #df.head(10)
17
18 #df.select_dtypes(include='object')
19
20 #df.select_dtypes(include='int64')
21
22 dfTransform=df.copy()
23
24
25 LE=LabelEncoder()
26 dfTransform['MaritalStatus']=LE.fit_transform(df['MaritalStatus'])
27 dfTransform['Gender']=LE.fit_transform(df['Gender'])
28
29 minmax=MinMaxScaler()
30
31 dfTransform['YearlyIncome']=df['YearlyIncome'].astype('float64',copy=False)
32 dfTransform['YearlyIncome']=pd.cut(df['YearlyIncome'],q=5,labels=0,1,2)
33 dfTransform['YearlyIncome']=minmax.fit_transform(dfTransform[['YearlyIncome']])
34
35 #totalchildren
36 dfTransform['TotalChildren']=df['TotalChildren'].astype('float64',copy=False)
37 dfTransform['TotalChildren']=minmax.fit_transform(dfTransform[['TotalChildren']])
38
39 #numberchildren at home
40 dfTransform['NumberChildrenAtHome']=df['NumberChildrenAtHome'].astype('float64',copy=False)
41 dfTransform['NumberChildrenAtHome']=minmax.fit_transform(dfTransform[['NumberChildrenAtHome']])
42
43 #number cars owned
44 dfTransform['NumberCarsOwned']=df['NumberCarsOwned'].astype('float64',copy=False)
45 dfTransform['NumberCarsOwned']=minmax.fit_transform(dfTransform[['NumberCarsOwned']])
46
```

```
-----
For NN classified with hidden layers=3 having (12, 10, 5) neurons
Test Accuracy:71.52131573252542
False positive rate:0.22436722436722437
Miss Rate:0.3462882096069869
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\network\_multilayer_perceptron.py:582: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
-----
For NN classified with hidden layers=3 having (50, 25, 12) neurons
Test Accuracy:75.26509413546852
False positive rate:0.18361218361218362
Miss Rate:0.31222707423580787
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\network\_multilayer_perceptron.py:582: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
-----
For NN classified with hidden layers=4 having (75, 25, 12, 5) neurons
Test Accuracy:75.02705042198659
False positive rate:0.2059202059202059
Miss Rate:0.294321441048035
Visualising 2D- PCA components
Visualising 5- PCA components
In [9]:
```

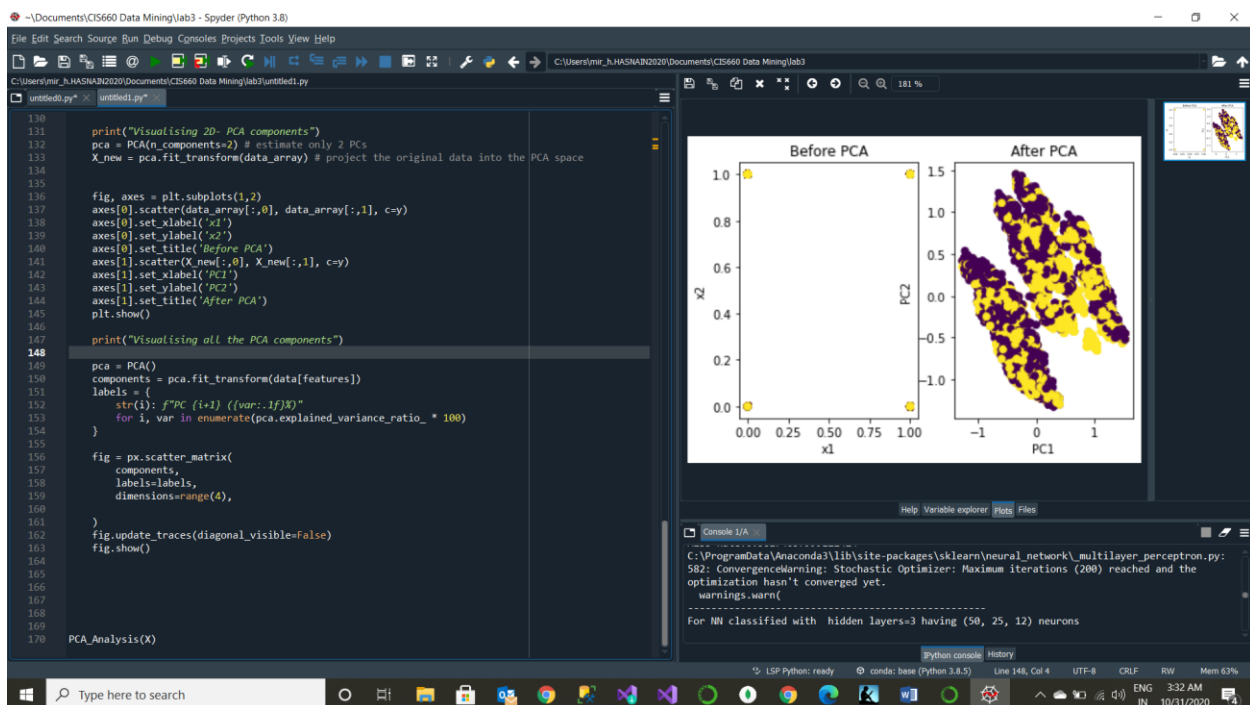
PART-2 3 Feature significance analysis with PCA tools (Extra Credit):

PCA technique is particularly useful in processing data where multi-collinearity exists between the features/variables.

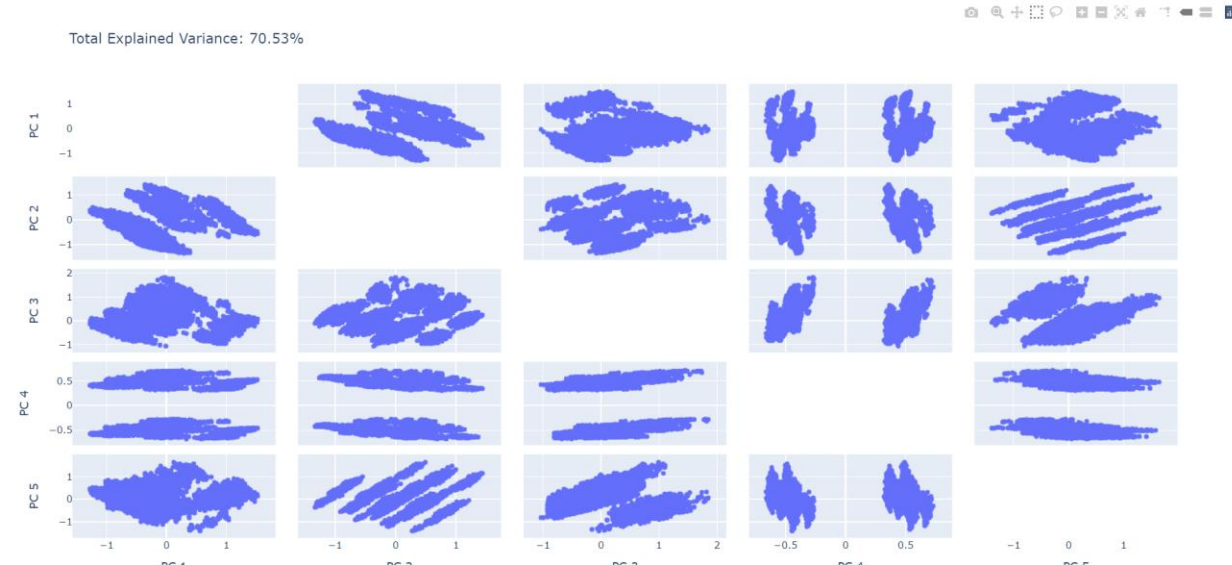
PCA analysis is done on 12 features of our pre-processed BikeBuyer dataset. The features selected are :

```
[ 'MaritalStatus', 'Gender', 'YearlyIncome', 'TotalChildren',  
  'NumberChildrenAtHome', 'EnglishEducation', 'EnglishOccupation',  
  'HouseOwnerFlag', 'NumberCarsOwned', 'CommuteDistance', 'Age',  
  'Region_Europe', 'Region_North America', 'Region_Pacific' ]
```

In the following we plot a 2D plot by doing a PCA analysis for feature selection on our features using number of PCA components **nPCA=2**



Total Explained Variance: 70.53%



PC 1

PC 2

PC 3

PC 4

PC 5

PC 1

PC 2

PC 3

PC 4

PC 5

1. SVM

1. For SVM classified with **kernel=linear**
Accuracy:63.83899588833586
False positive rate:0.30848329048843187
Miss Rate:0.41582859641451686

2. For SVM classified with **kernel=rbf**
Accuracy:71.2832720190435
False positive rate:0.23221936589545844
Miss Rate:0.34324442501093133

3. For SVM classified with **kernel=poly**
Accuracy:71.71607877082883
 False positive rate:0.2467866323907455
 Miss Rate:0.31963270660253607

2. Neural Network classifier :

The highest accuracy is with the Neural network topology having hidden layers=3 having (50, 25, 12) units, followed by a very close margin by the neural network having hidden layers=4 having (75, 25, 12, 5) units.

Thus we see that after a point the increase in number of layers doesn't impact the accuracy much and counter intuitively adding too many layers and units could increase the execution time but not increase the accuracy or might even reduce it.

For NN classified with

1. hidden layers=3 having (12, 10, 5) neurons

Accuracy:71.9974031594893

False positive rate:0.23005487547488393

Miss Rate:0.33259325044404975

2. hidden layers=3 having (50, 25, 12) neurons

Accuracy:75.89266392555723

False positive rate:0.16040523427606584

Miss Rate:0.3259325044404973

3. hidden layers=4 having (75, 25, 12, 5) neurons

Accuracy:75.80610257520017

False positive rate:0.25707049387927394

Miss Rate:0.22602131438721138

PART -4

- NN have better accuracy than SVM on the same data because SVM uses only a subset of a dataset as training data thus the number of observations required to train an SVM isn't high whereas a NN uses the entire dataset.
- SVM works best when the preprocessed input data is vectorized whereas Neural networks work best with normalized data usually between [0,1]
- The highest accuracy of 75.89 % on the same dataset was yielded by the Neural network classifier with hidden layers=3 having (50, 25, 12) units in the layers respectively.
- The lowest accuracy 63.8% is found in the SVM classifier trained with kernel=linear, however the accuracy with rbf is much better, this is because rbf is better suited to handle data with outliers.
- The highest accuracy is with the Neural network topology having hidden layers=3 having (50, 25, 12) units, followed by a very close margin by the neural network having hidden layers=4 having (75, 25, 12, 5) units.
- after a point the increase in number of layers in the Neural Network didn't increase the accuracy of the BikeBuyer dataset much
- NN yielded better results when the optimizer used was Adam or lbfgs instead of sgd on the bike buyer dataset.

CODE:

The code is in 2 files :

1. Datatransformation.py

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, MinMaxScaler

df=pd.read_csv('VTargetBuyers.csv')
df.drop(columns=['CustomerKey','GeographyKey'],inplace=True)
yLabels=df[['BikeBuyer']]
df.drop(columns=['BikeBuyer'],inplace=True)
dfTransform=df.copy()

LEncoder=LabelEncoder()
dfTransform['MaritalStatus']=LEncoder.fit_transform(df['MaritalStatus'])
dfTransform['Gender']=LEncoder.fit_transform(df['Gender'])

minmax=MinMaxScaler()

dfTransform['YearlyIncome']=df['YearlyIncome'].astype('float64',copy=False)
dfTransform['YearlyIncome']=pd.qcut(df['YearlyIncome'], q=5,labels=[0,1,2,3,4])
dfTransform['YearlyIncome']=minmax.fit_transform(dfTransform[['YearlyIncome']])

#totalchildren
dfTransform['TotalChildren']=df['TotalChildren'].astype('float64',copy=False)
```

```
dfTransform['TotalChildren']=minmax.fit_transform(dfTransform[['TotalChildren']])
```

```
#numberchildren at home
```

```
dfTransform['NumberChildrenAtHome']=df['NumberChildrenAtHome'].astype('float64',copy=False)
```

```
dfTransform['NumberChildrenAtHome']=minmax.fit_transform(dfTransform[['NumberChildrenAtHome']])
```

```
#number cars owned
```

```
dfTransform['NumberCarsOwned']=df['NumberCarsOwned'].astype('float64',copy=False)
```

```
dfTransform['NumberCarsOwned']=minmax.fit_transform(dfTransform[['NumberCarsOwned']])
```

```
dfTransform['CommuteDistance']=df['CommuteDistance'].replace(['0-1 Miles','1-2 Miles','2-5 Miles', '5-10 Miles', '10+ Miles'],[0,1,2,3,4])
```

```
dfTransform['CommuteDistance']=minmax.fit_transform(dfTransform[['CommuteDistance']])
```

```
#English education
```

```
dfTransform['EnglishEducation']=df['EnglishEducation'].replace(['Partial High School','High School','Partial College', 'Bachelors', 'Graduate Degree'],[0,1,2,3,4])
```

```
dfTransform['EnglishEducation']=minmax.fit_transform(dfTransform[['EnglishEducation']])
```

```
#English occupation
```

```
dfTransform['EnglishOccupation']=df['EnglishOccupation'].replace(['Manual','Skilled Manual','Clerical','Professional', 'Management'],[0,1,2,3,4])
```

```
dfTransform['EnglishOccupation']=minmax.fit_transform(dfTransform[['EnglishOccupation']])
```

```
OHE=pd.get_dummies(df['Region'],prefix='Region')
```

```
dfTransform=dfTransform.drop('Region',axis=1)
```

```
dfTransform=dfTransform.join(OHE)
```

```
dfTransform['Age']=pd.qcut(df['Age'], q=5,labels=[0,1,2,3,4])
dfTransform['Age']=minmax.fit_transform(dfTransform[['Age']])

#dfTransform.head(12)

#joining class labels to the transformed df
dfTransform=dfTransform.join(yLabels)

dfTransform.to_csv('NormalizedData.csv',index=False)
```

2. lab3.py

```
import pandas as pd
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from numpy import absolute,mean,std

df=pd.read_csv('NormalizedData.csv')

y=df['BikeBuyer'].ravel()
X=df.drop(columns=['BikeBuyer'])
```

```
X_train, X_test, y_train, y_test = train_test_split( X, y)
```

```
def SVM_Classification(TrainingData,TrainLabels,TestData,TestLabels,k='linear'):
```

```
    from sklearn.metrics import confusion_matrix
```

```
    Train=np.asarray(TrainingData)
```

```
    Labels=np.asarray(TrainLabels)
```

```
    Test=np.asarray(TestData)
```

```
    TestLabels=np.asarray(TestLabels)
```

```
    clf=SVC(kernel=k,gamma='scale')
```

```
    clf.fit(Train,Labels)
```

```
    prediction=clf.predict(Test)
```

```
    #true positve, true negative, false positive, false negative
```

```
    tn, fp, fn, tp = confusion_matrix(TestLabels, prediction).ravel()
```

```
    #accuracy
```

```
    acc=(tp+tn)/(tp+tn+fp+fn)
```

```
    acc=acc*100
```

```
    #false positive rate & miss rate
```

```
    fpr=fp/(fp+tn)
```

```
    missrate=fn/(tp+fn)
```

```
print("-----")
print("For SVM classified with kernel={}".format(k))
#print("Cross Validation of Train Dataset ")
#scores = cross_val_score(clf, X, y, cv=5)
# summarize the model performance
#print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
print("\nTest Data")
print("\n Test Accuracy:{}".format(acc))
print("False positive rate:{}".format(fpr))
print("Miss Rate:{}".format(missrate))
```

```
return acc,fpr,missrate
```

```
#trainig SVM with linear kernel
```

```
acc,fpr,missrate=SVM_Classification( X_train, y_train, X_test, y_test)
```

```
#training the SVM with kernel=rbf
```

```
acc,fpr,missrate=SVM_Classification(X_train, y_train, X_test, y_test,k='rbf')
```

```
#training the SVM with kernel=polynomial
```

```
acc,fpr,missrate=SVM_Classification(X_train, y_train, X_test, y_test,k='poly')
```

```

def NN_Classification(TrainingData,TrainLabels,TestData,TestLabels,hiddenLayerTuple=(12,10,5,2)):

    from sklearn.metrics import confusion_matrix

    Train=np.asarray(TrainingData)
    Labels=np.asarray(TrainLabels)

    Test=np.asarray(TestData)
    TestLabels=np.asarray(TestLabels)

    clf=MLPClassifier(solver='adam', alpha=1e-5, hidden_layer_sizes=hiddenLayerTuple,
random_state=1)

    clf.fit(Train,Labels)

    prediction=clf.predict(Test)

    #true positive, true negative, false positive, false negative
    tn, fp, fn, tp = confusion_matrix(TestLabels, prediction).ravel()

    #accuracy
    acc=(tp+tn)/(tp+tn+fp+fn)
    acc=acc*100

    #false positive rate & miss rate
    fpr=fp/(fp+tn)
    missrate=fn/(tp+fn)

    print("-----")

    print("For NN classified with hidden layers={} having {}
neurons".format(len(hiddenLayerTuple),hiddenLayerTuple))

```



```
#print("Cross Validation of Train Dataset ")

#scores = cross_val_score(clf, X, y, cv=5)

# summarize the model performance

#print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

print("\n Test Accuracy:{}".format(acc))

print("False positive rate:{}".format(fpr))

print("Miss Rate:{}".format(missrate))
```

```
return acc,fpr,missrate
```

```
#trainig NN with 4 layers
```

```
acc,fpr,missrate=NN_Classification( X_train, y_train, X_test, y_test,hiddenLayerTuple=(12,10,5))
```

```
#training the NN with 4 layers
```

```
acc,fpr,missrate=NN_Classification(X_train, y_train, X_test, y_test,hiddenLayerTuple=(50,25,12))
```

```
#training the NN with 4 layers
```

```
acc,fpr,missrate=NN_Classification(X_train, y_train, X_test, y_test,hiddenLayerTuple=(75,25,12,5))
```

```
def PCA_Analysis(data):
```

```
    import matplotlib.pyplot as plt
```

```
    from sklearn.decomposition import PCA
```

```
    import plotly.express as px
```

```
    from plotly.offline import plot
```

```
features=list(data.columns)
data_array=np.asarray(data)

print("Visualising 2D- PCA components")
pca = PCA(n_components=2) # estimate only 2 PCs
X_new = pca.fit_transform(data_array) # project the original data into the PCA space
```

```
fig, axes = plt.subplots(1,2)
axes[0].scatter(data_array[:,0], data_array[:,1], c=y)
axes[0].set_xlabel('x1')
axes[0].set_ylabel('x2')
axes[0].set_title('Before PCA')
axes[1].scatter(X_new[:,0], X_new[:,1], c=y)
axes[1].set_xlabel('PC1')
axes[1].set_ylabel('PC2')
axes[1].set_title('After PCA')
plt.show()
```

```
print("Visualising 5 PCA components")
```

```
pca = PCA(n_components=5)
components = pca.fit_transform(data[features])
total_var = pca.explained_variance_ratio_.sum() * 100
labels = {str(i): f"PC {i+1}" for i in range(5)}
```

```
fig=px.scatter_matrix(
    components,
    labels=labels,
```

```
        dimensions=range(5),  
        title=f'Total Explained Variance: {total_var:.2f}%',  
    )  
    fig.update_traces(diagonal_visible=False)  
    fig.show()  
    plot(fig)  
PCA_Analysis(X)
```