Building Machine Learning Models for the Prediction of Dependent Variables in the Given Datasets

Assignment 3

Machine Learning

IOT Track

Muhammad Farrukh Mehmood

Reg no. 399602 Fall/22

Table of Contents

List of Figures

List of Tables

The objective of the assignment is to develop support vector machine and k nearest neighbors (KNN) classificiation models to predict the dependent variables based on the independent variables in the following datasets:

I. Zoo data set (SVM)
II. Social media ads dataset (KNN)

1.1 Road map

To achieve the target following procedural steps were adopted:

- Data cleaning
- Data visualization
- Dividing the data into the label(dependent variable) and features (independent variables)
- Splitting the data into training, validation and test sets with the proportion of 60,20,20 respectively
- Finding an appropriate model with suitable hyperparameters
- Training the model with training data
- Validating the model on train and test sets with different scoring parameters

1.2 Exploratory Data Analysis

Firstly, the data were checked for missing or null values. To quantify the null values "isnull().any()" command was used. The command reported no null values in both datasets.

```
# zoo data set missing value check
df.isnull().any()

animal_name    False
hair           False
feathers       False
eggs           False
milk           False
airborne       False
aquatic        False
predator       False
toothed        False
backbone       False
breathes       False
venomous       False
fins           False
legs           False
tail           False
domestic       False
catsize        False
class_type     False
dtype: bool
```

```
# social media ads missing value check
df.isnull().any()

Age                False
EstimatedSalary    False
Purchased          False
Gender_Male        False
dtype: bool
```

Figure 1: Missing value report for both datasets

To visualize the missing values, seaborn's heatmap function was used.

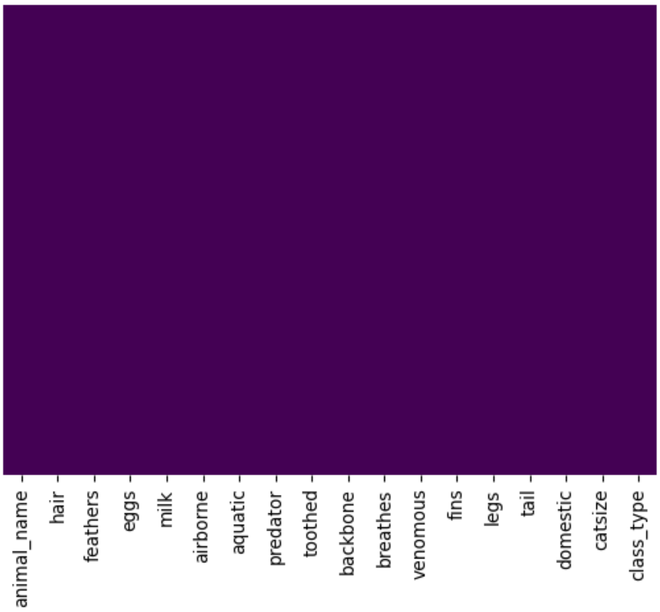

Figure 2: Missing Value chart of zoo data set

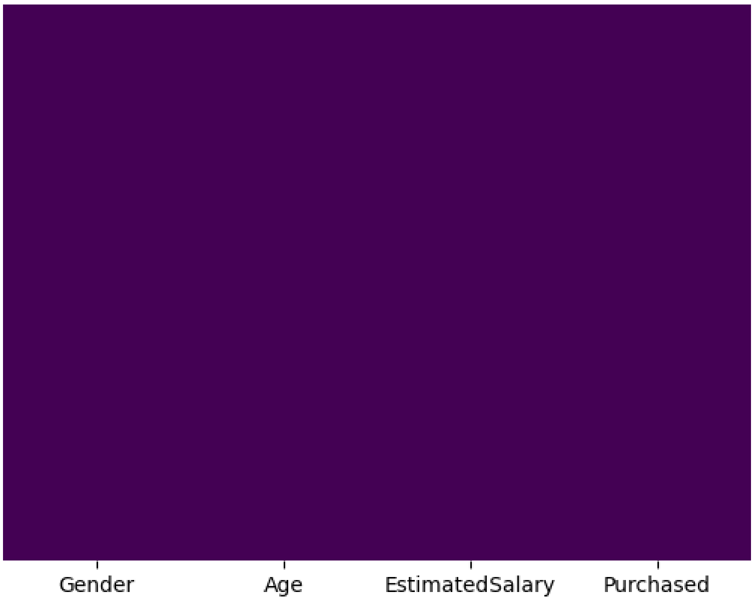

Figure 3: Missing Value chart of Social media ads data set

1.3 Value Counts for Zoo datasets

The following chart shows the number of animals correspoinding to different classes. The classes are represented numerically.

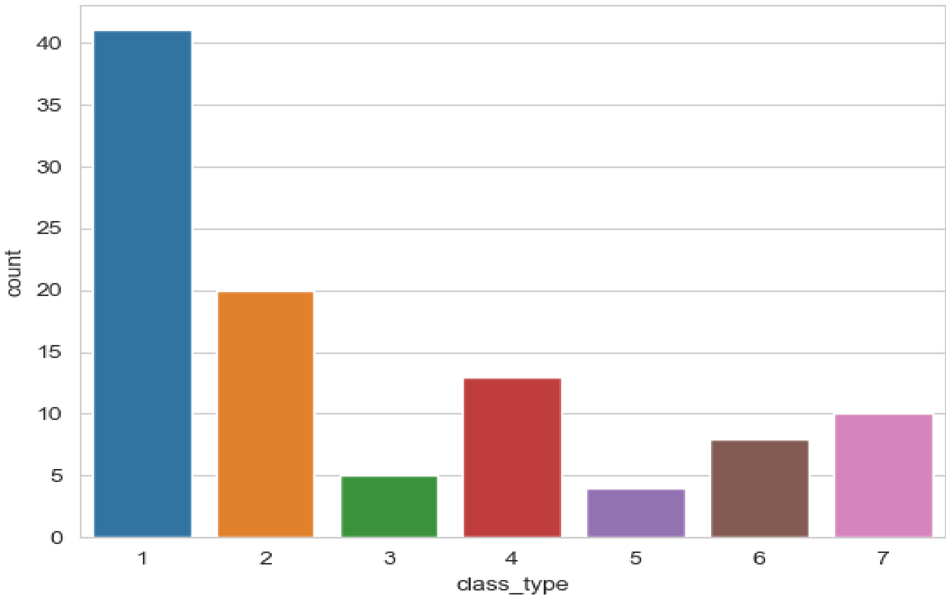

Figure 4: Class distribution of zoo dataset

The following chart shows the legs-wise distribution of classes.

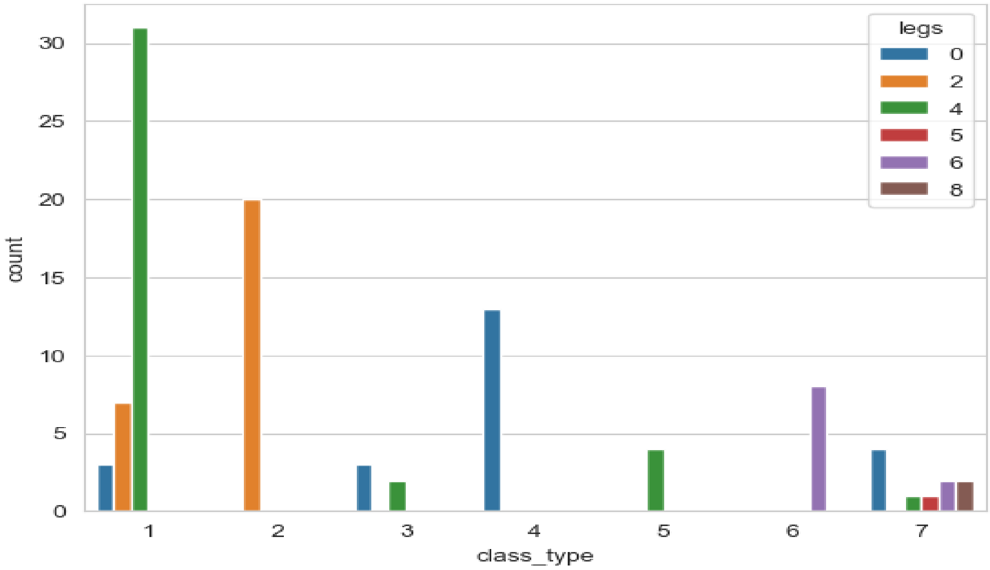

Figure 5: Class distribution based on number of legs

Distribution of animal classes which produce and do not produce the milk:

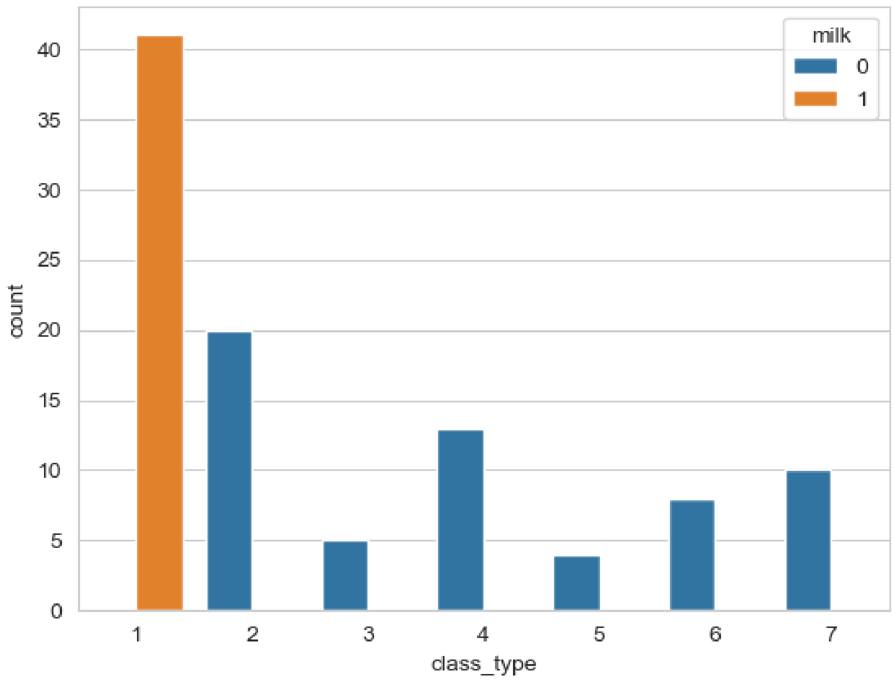

Figure 6: Class distribution based on milk

1.4 Value Counts for Social media ads datasets

The following chart shows the distribution of purchase status.

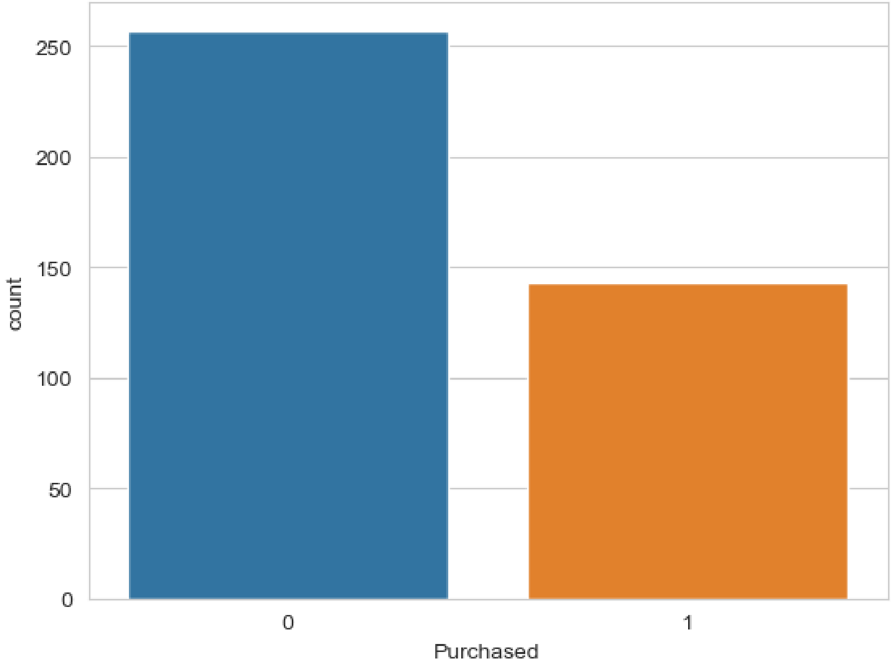

Figure 7: Distribution of purchase status

The following chart shows the gender-wise distribution of purchase status. We can see purchasing is done mostly by females.

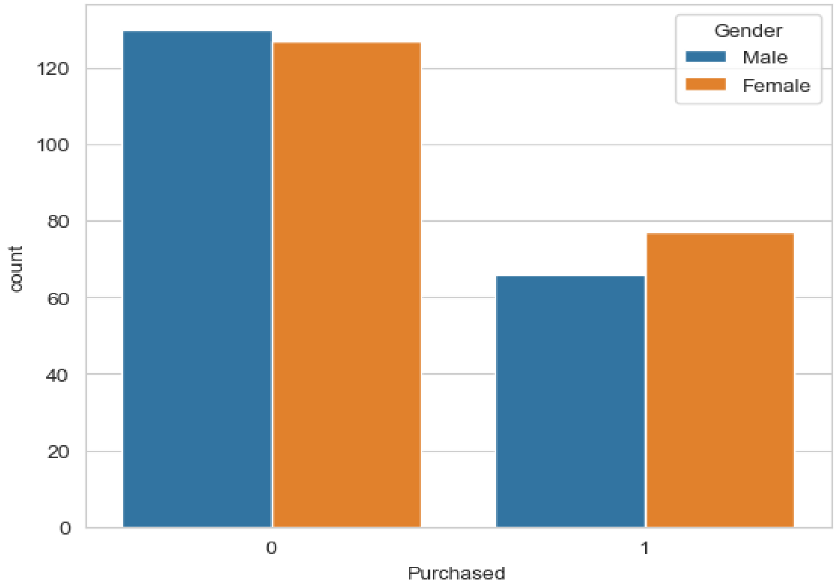

Figure 8: Distribution of purchase status based on gender

1.5 Correlation charts

Following charts show the correlation between attributes of datasets:

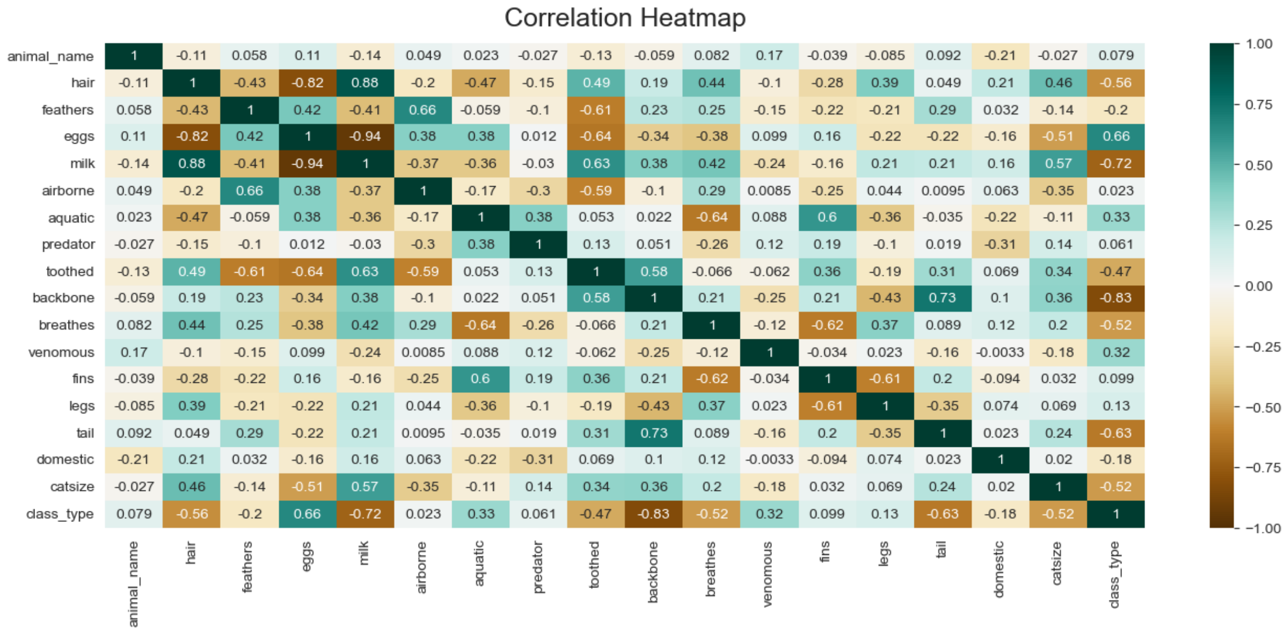

Figure 9: Correlation between attributes of zoo dataset

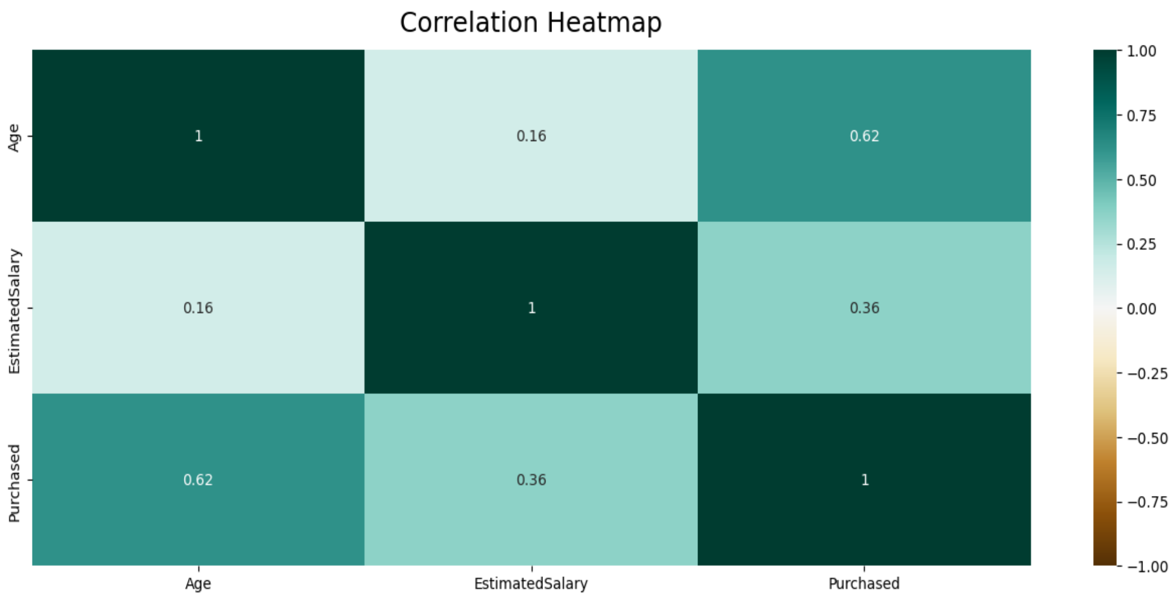

Figure 10: Correlation between attributes of Social media ads dataset

1.6 Converting the Categorical Data into numerical

Categorical data were converted to numerical data through One Hot encoding.

- Dummy variables were created corresponding to the categorical attributes using ‘get_dummies()’ command
- These dummy variables were concatenated with the actual data set.
- The actual categorial attributes columns were dropped off from the data set to get pure numerical dataset

The implementation of the above procedure can be seen in the code. The data were normalized based on StandardScaler scaling technique.

1.7 Dividing the Actual data into ‘feature’ and ‘label’ datasets

The actual datasets were divided into two new data sets. All the independent variables were assigned a dataset ‘feature’ and all the single dependent variable was assigned the dataset ‘label’.

1.8 Splitting the data into test and training data set

For model building and validation, the data were split into three parts (Zoo data): 60 percent data for model training, 20 percent data for model validation and 20 percent for model testing. For this purpose, ‘train_test_split()’ function was used from ‘sklearn’ library.

Social media ads data were split into train and test with 80-60 proportion.

2 Training the model on Zoo data set

2.1 Support Vector Machine with self-written code

Since the dataset contained 7 classes, SVM classification was implemented using a one-vs-all approach, first with a self-written code 2^{nd} with sklearn's built-in modules. Following functions were defined in self written code:

- Linear kernel function
- Cost function
- Gradient Descent
- Functions for prediction

2.1.1 Results of self-written code

Hyper parameters:

Table 1: SVM hyper parameters (self -written code)

Hyper parameter	**Value**
Learning Rate	0.01
No. of Iterations	1000
Normalization Technique	Standard Scaler
Lambda parameter	0.01

Based on above hyper paratmeters, following accuracies were achieved:

Table 2: Score of self-written code

Train Accuracy	0.9167
Validation Accuracy	0.75
Test Accuracy	0.9524

2.2 SVM with Sklearn library

Sklearn’s SVM model was trained on the dataset against following 3 kernel functions.

- RBF kernel
- Polynomial kernel
- Sigmoid kernel

The accuracies on each of the kerneal function were calculated against different values of gamma and regularization parameters and plotted on graphs.

Accuracy plots of RBF kernel

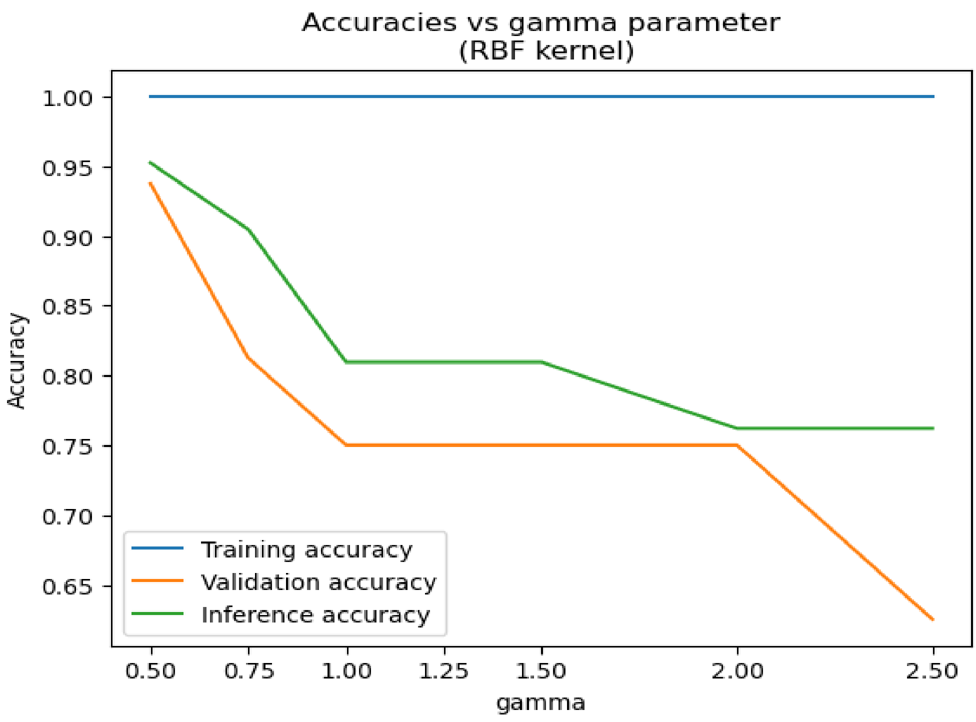

Figure 11: Accuracy vs gamma parameter (RBF kernel)

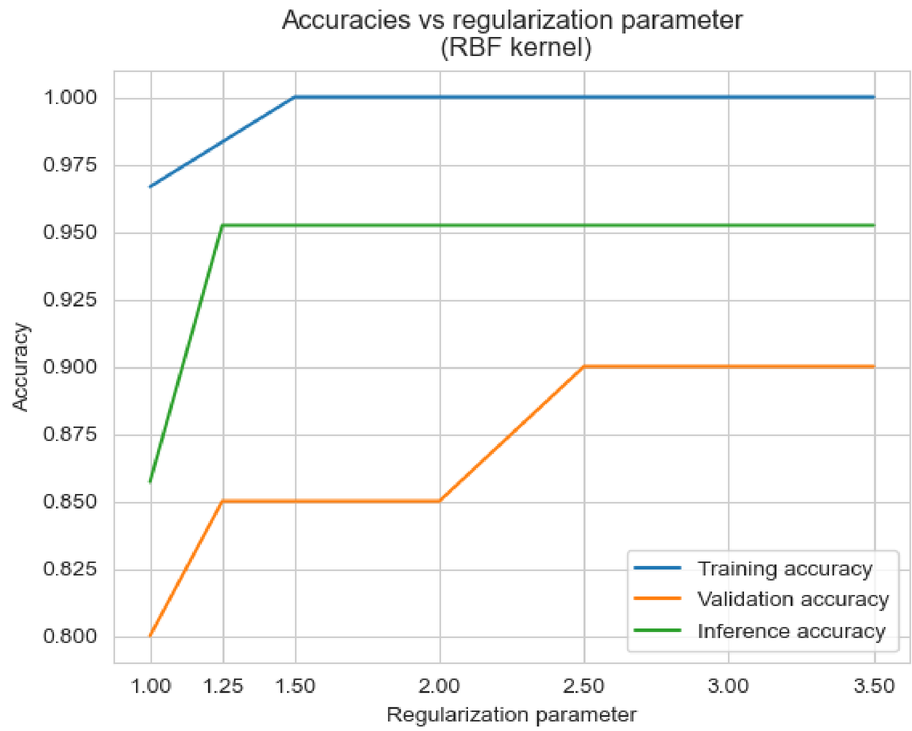

Figure 12: Accuracy vs regularization parameter (RBF kernel)

Accuracy plots of poly kernel

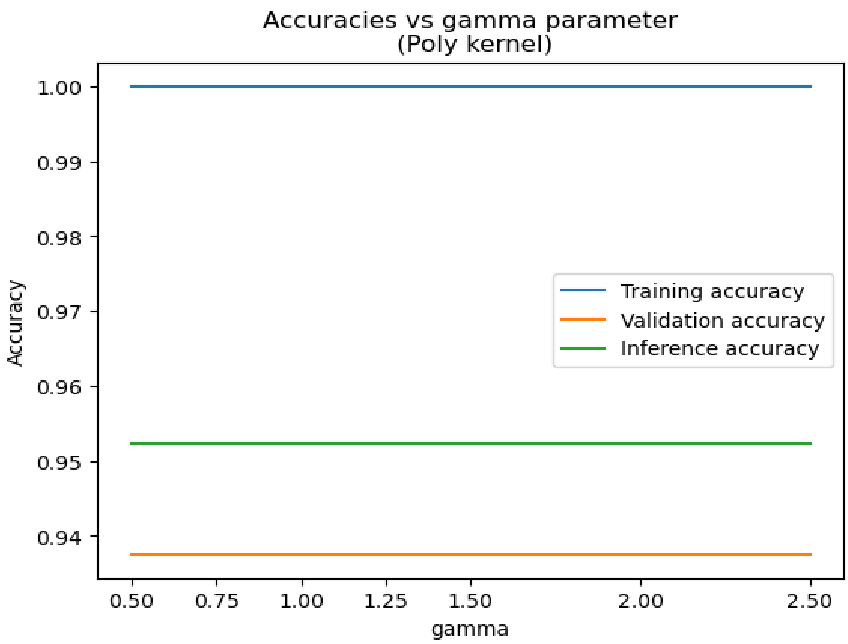

Figure 13: Accuracy vs gamma parameter (Polynomial kernel)

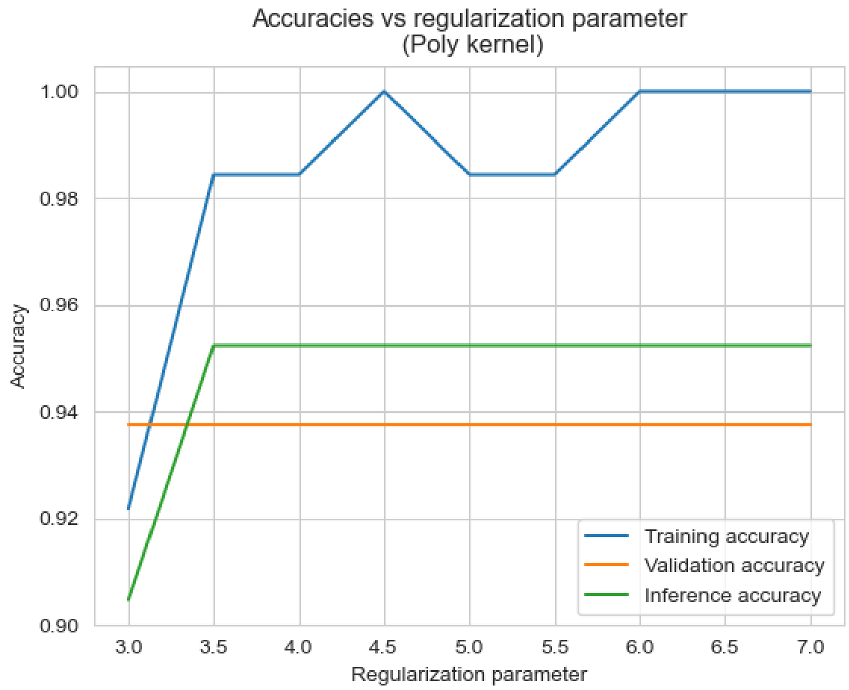

Figure 14: Accuracy vs regularization parameter (Polynomial kernel)

Accuracy plots of sigmoid kernel

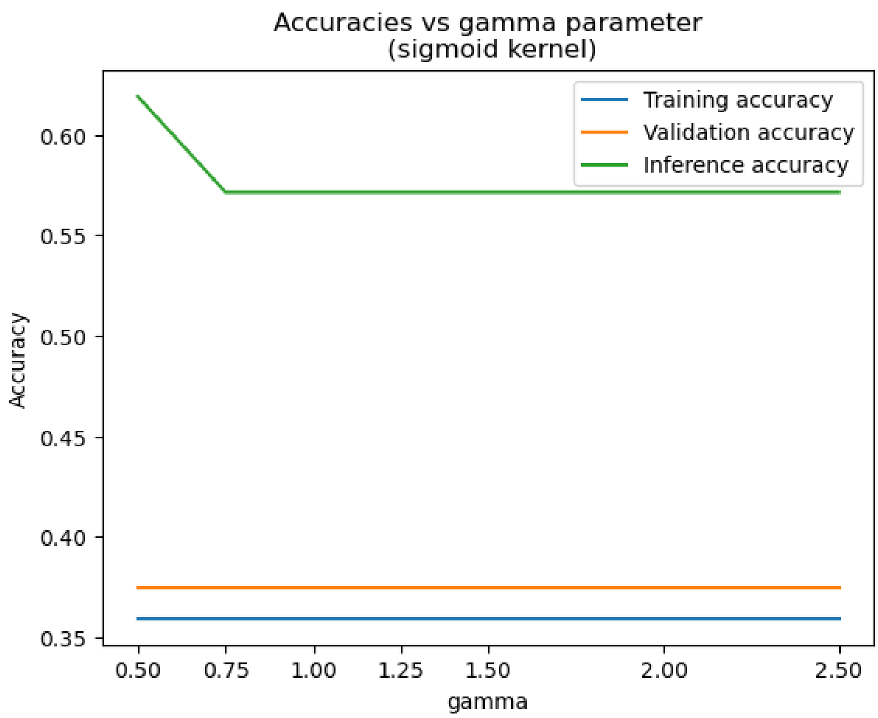

Figure 15: Accuracy vs gamma parameter (Sigmoid kernel)

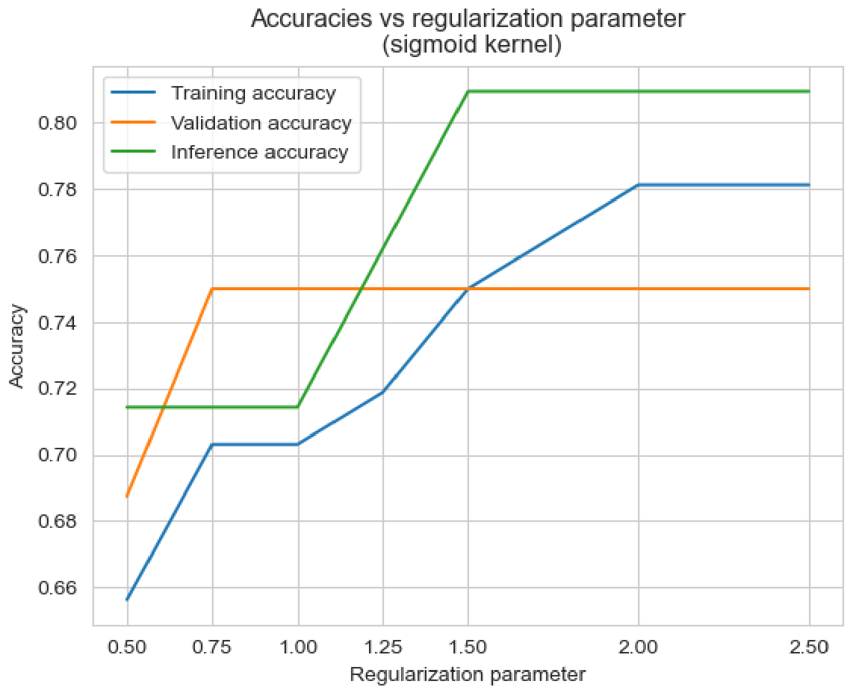

Figure 16: Accuracy vs regularization parameter (Sigmoid kernel)

2.3 Discussion on Plots

RBF kernel:

Training accuracy does not get affected by varying gamma values. Validation and inference accuracies decreases by increasing the gamma. Inference accuracy achieves plateau value at gamma of value 2.0. All the accuracies achieve plateau value at regularization value of 1.50. Inference accuracy remains below the training accuracy

Polynomial kernel:

Gamma parameter has no affect on any of the accuracies. Inference accuracy remains below the training accuracy.

Sigmoid kernel:

In case of sigmoid kernel, inference accuracy remains higher than training accuracy for both the parameters.

2.4 Hyper parameter tunning through GridSearchCV

The model was tuned for three hyperparameters (kernel function, gamma values and regularization parameter) using GridSearchCV. GridSearchCV provided following values of hyperparameters as optimum:

Table 3: SVM Hyper parameters

Hyper parameter	Value
Kernel function	Polynomial
Gamma	0.5
Regularization parameter	0.5

2.4.1 Score

Table 4: Model accuracy

Train Accuracy	1.0
Test Accuracy	0.9524

2.5 Some correctly and incorrectly classified examples

Table 5: Correct and incorrect classifications (zoo dataset)

hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	Predicted Class	Actual Class
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4	4
0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	4	4
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4	4
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	7	1
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	7	1
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	7	1

3 Training the model on Social media ads data set

3.1 K nearest neighbors classification

Since the dataset contained binary class, Sklearn's KNearestnegihbors module was trained on the given data set. The performance of the model was tested against different values of a single hyper parameter namely number of nearest neighbors.

Accuracy plots of KNN against different values of nearest neighbors

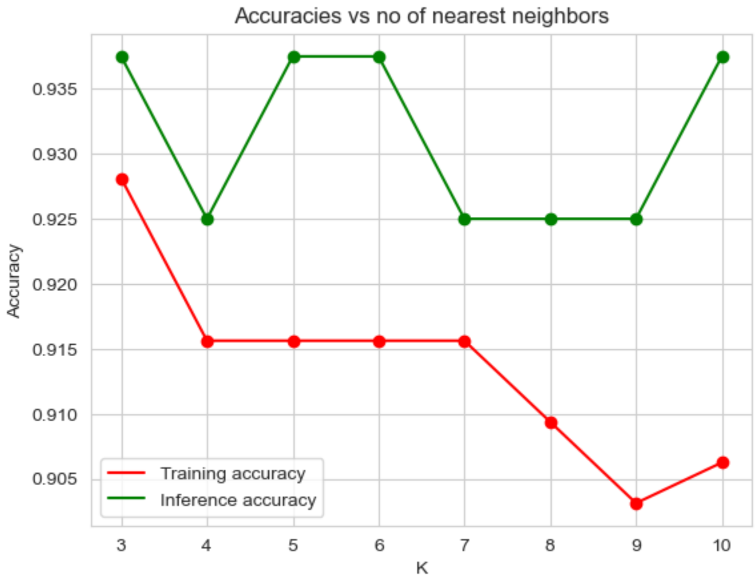

Figure 17: Training and inference accuracy of KNN model

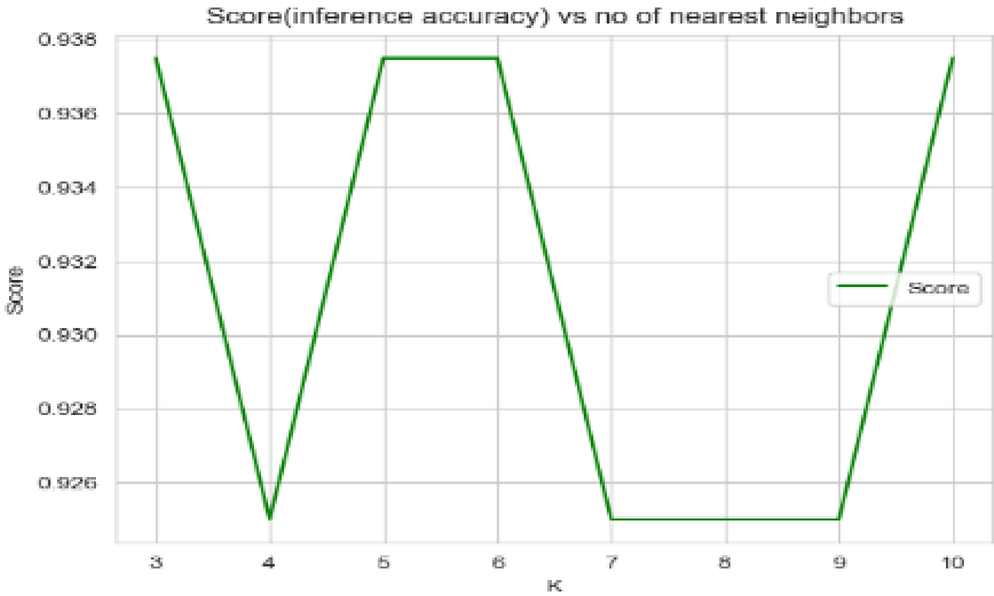

Figure 18: Score of model

3.2 Discussion on Plots

The inference accuracy fluctuates between 0.925 and 0.938. It always remains higher than the training accuracy.

3.3 Hyper parameter tunning through GridSearchCV

Sklearn's GridSearchCV module was implemented to obtain optimum number of nearest numbers. It yielded 9 nearest neighbors. It was found that the model worked fine if the nearest neighbors were kept 3 instead of 9.

3.4 Decision boundary

The dataset contained 3 features and binary class. The decision boundary can be visualized by 4 different plots given following:

1. Taking all the features at a time (3D plot)

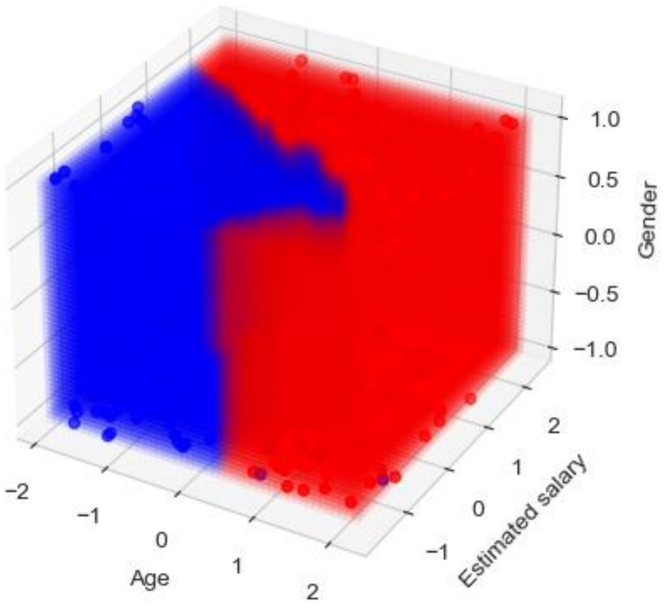

Figure 19: Decision boundary taking all the features at a time

2. Decision boundary when taking "Age" and "Estimated Salary" (2D plot):

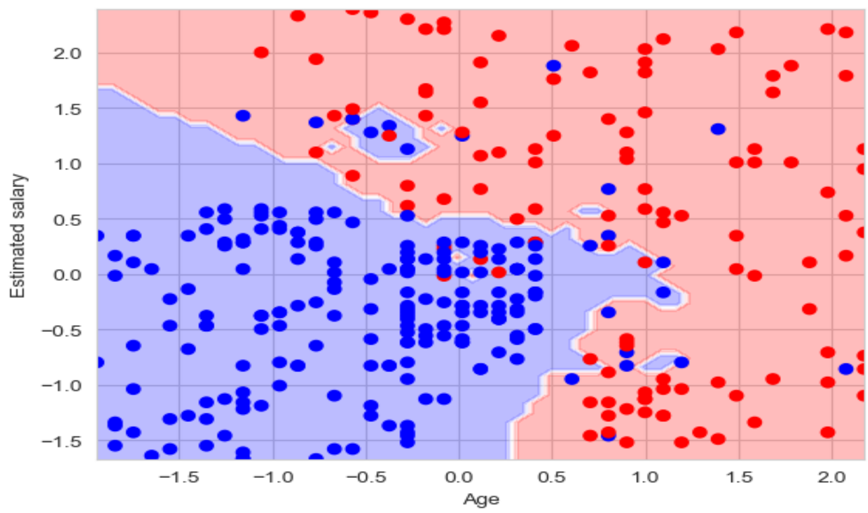

Figure 20: Decision boundary when taking "Age" and "Estimated Salary"

3. Decision boundary when taking “Gender” and “Estimated Salary” (2D plot):

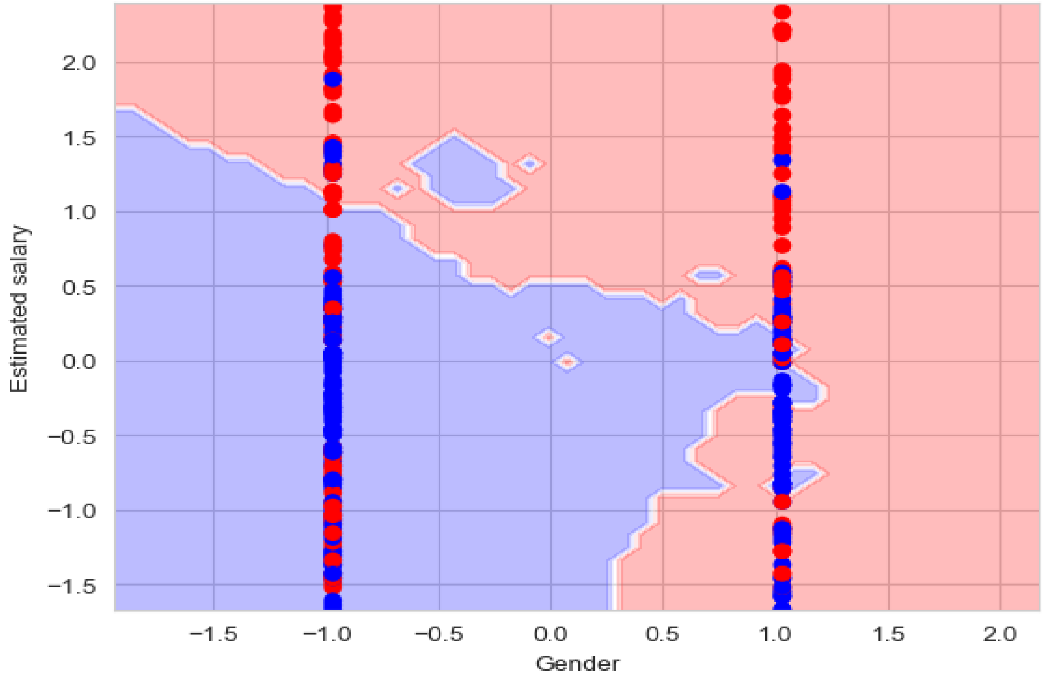

Figure 21: Decision boundary when taking “Gender” and “Estimated Salary”

4. Decision boundary when taking “Age” and “Gender” (2D plot):

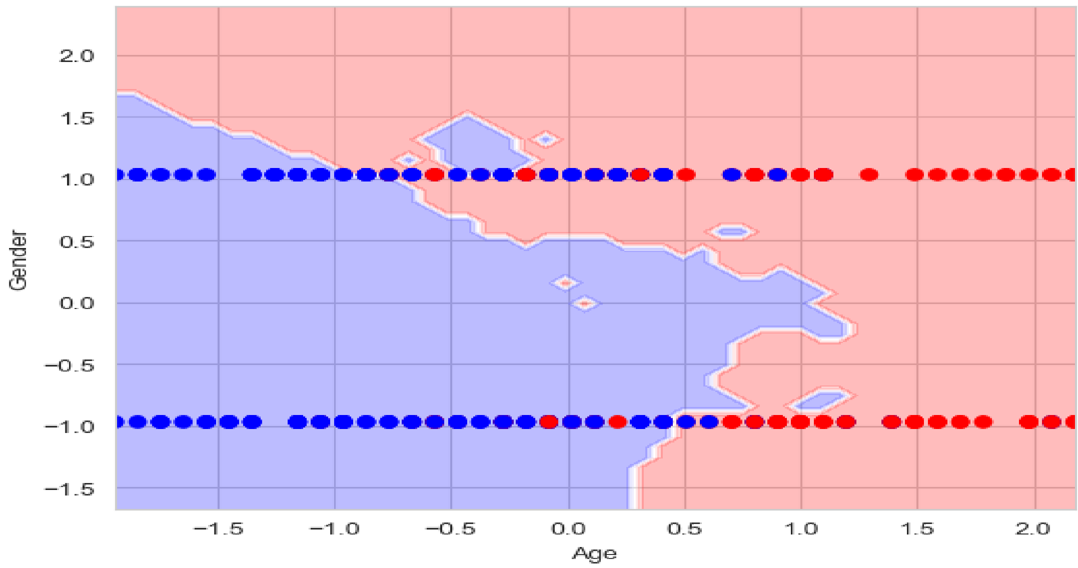

Figure 22: Decision boundary when taking “Age” and “Gender”

3.5 Some correctly and incorrectly classified examples

Table 6: Correct and incorrect classifications (Social media ads dataset)

Age	EstimatedSalary	Gender	Purchase prediction	Purchase actual
33	51000	Female	0	0
32	120000	Male	1	1
20	23000	Female	0	0
35	108000	Male	1	0
59	83000	Female	1	0
33	113000	Female	1	0