Social Network Ads prediction

Revanth Reddy Bandaru **Date:12-09-2021**

Table of Contents:

Problem definition:

How we can use various python-based Machine Learning Model to and the given parameters to predict the if the ad is shown will the item be purchase. In simpler words we tell whether a user on Social Networking site after clicking the ad's displayed on the website, end's up buying the product or not. This could be helpful for the company selling the product.

Let's say that it's a car company which has paid the social networking site (For simplicity we'll assume its Facebook from now on)to display ads of its newly launched car. Now since the company relies heavily on the success of its newly launched car it would leave no stone unturned while trying to advertise the car.

Well then what's better than advertising it on the most popular platform right now. But what if we only advertise it to the correct crowd. This could help in boosting sales as we will be showing the ad of the car only to selected crowd.

So, this is where you come in…

The Car company has hired you as a Data Scientist to find out the correct crowd, to which you need to advertise the car and find out the people who are most likely to buy the car based on certain features which describe the type of users who have bought the car previously by clicking on the ad.

Data:

Data set from: https://www.kaggle.com/d4rklucif3r/social-network-ads

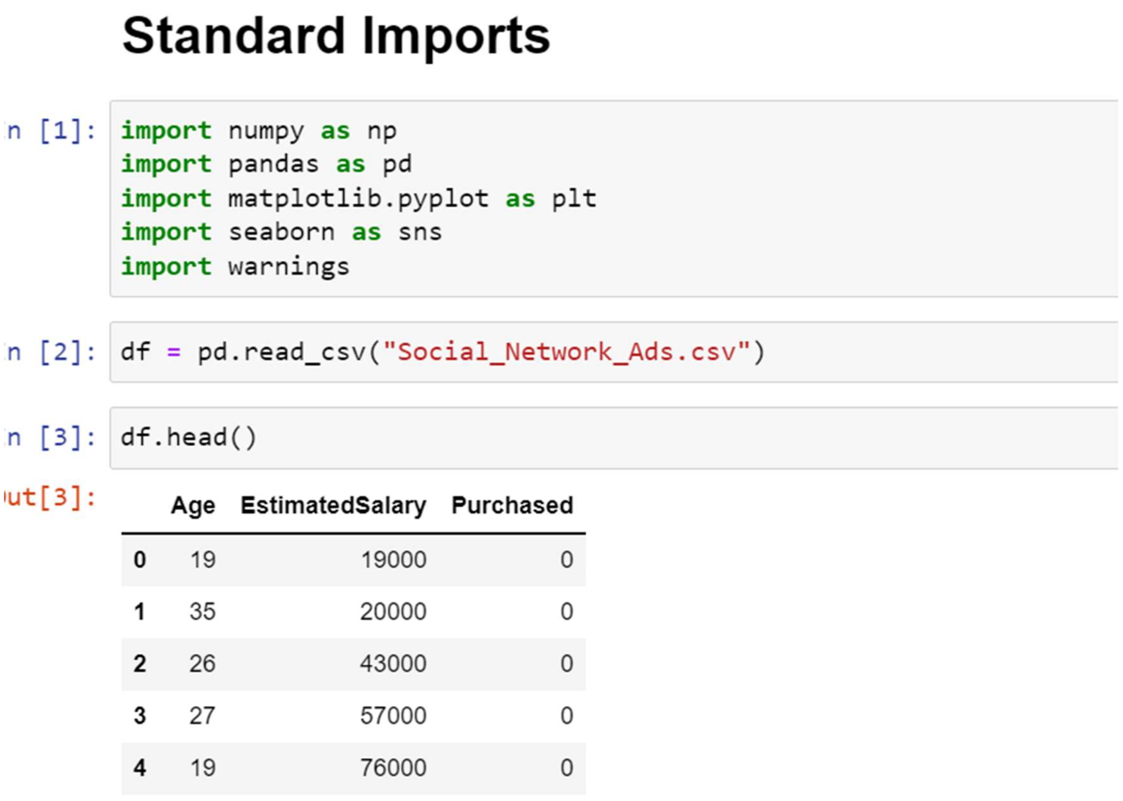
The Dataset used in these models talks about whether a person of certain age having certain income purchases a product or not. We need to predict whether a targeted audience will purchase the product or not.

The Dataset contains information about users on a Social Networking site and using that info as Features for our ML model, we try to predict that whether a particular user after clicking on a ad on the Social networking site goes on to buy a particular product or not.

Well, this particular Social Network has a business client which as mentioned earlier is a car company which advertises itself by putting adds on the social networking site. Now the work of the social network here is to gather information as to whether the user bought the product or not. The dependent variable in this case is Purchased which is 1 if user purchases the car and 0 otherwise.

So, the goal here is to create a classifier which would put each user into the correct category by predicting as to whether he's buying the product or not.

Pre-Processing:



Standard Imports

```
n [1]: import numpy as np
       import pandas as pd
       import matplotlib.pyplot as plt
       import seaborn as sns
       import warnings
```

```
n [2]: df = pd.read_csv("Social_Network_Ads.csv")
```

```
n [3]: df.head()
```

ut[3]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

The following features will be considered as the independent variables…

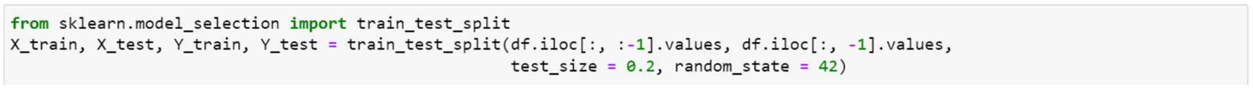
1) Age

2) Estimated Salary

Splitting the dataset into the Training set and Test set:

Importing the Cross Validation library which is now known as Model selection in newer versions of Python.

We divide the data into 75% data for training and 25% for testing our data



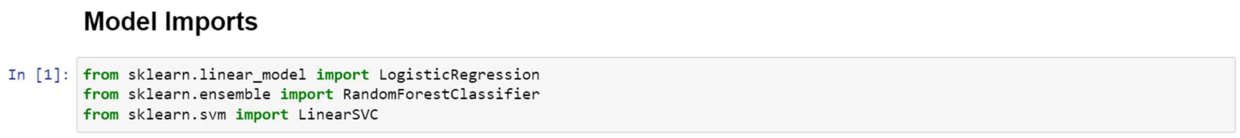
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(df.iloc[:, :-1].values, df.iloc[:, -1].values,
                                                    test_size = 0.2, random_state = 42)
```

Fitting Models:

Logistic regression : Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable have only two possible classes.

Random Forest Classifier : Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. It performs better results for classification problems.

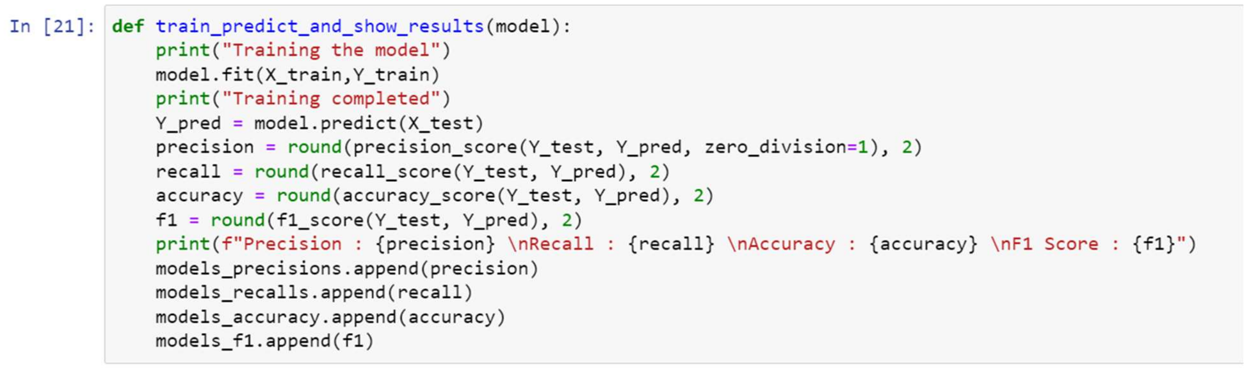
SVM(support vector Machines) : SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.



Model Imports

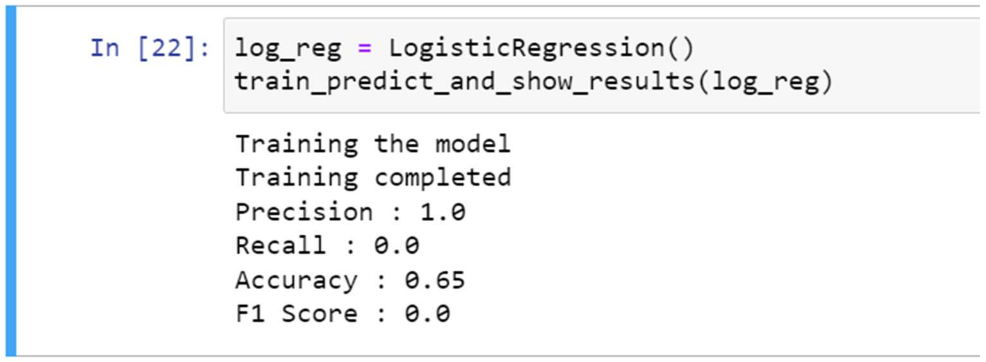
```
In [1]: from sklearn.linear_model import LogisticRegression
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.svm import LinearSVC
```

Fitting the models and finding precisions, recall score, model accuracy and f1 score using the below function.



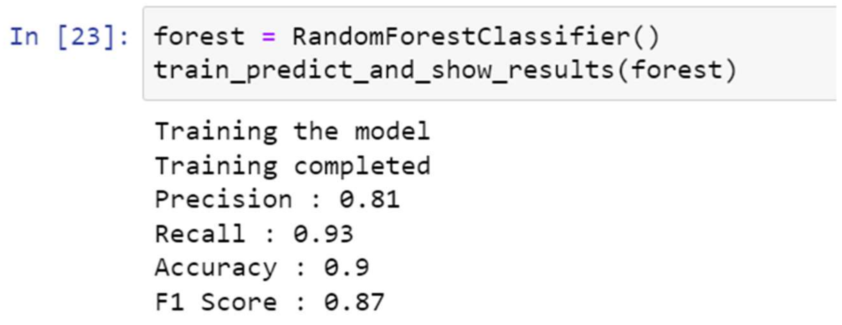
```
In [21]: def train_predict_and_show_results(model):
             print("Training the model")
             model.fit(X_train,Y_train)
             print("Training completed")
             Y_pred = model.predict(X_test)
             precision = round(precision_score(Y_test, Y_pred, zero_division=1), 2)
             recall = round(recall_score(Y_test, Y_pred), 2)
             accuracy = round(accuracy_score(Y_test, Y_pred), 2)
             f1 = round(f1_score(Y_test, Y_pred), 2)
             print(f"Precision : {precision} \nRecall : {recall} \nAccuracy : {accuracy} \nF1 Score : {f1}")
             models_precisions.append(precision)
             models_recalls.append(recall)
             models_accuracy.append(accuracy)
             models_f1.append(f1)
```

Using above defined function we call each respective model to find the following.



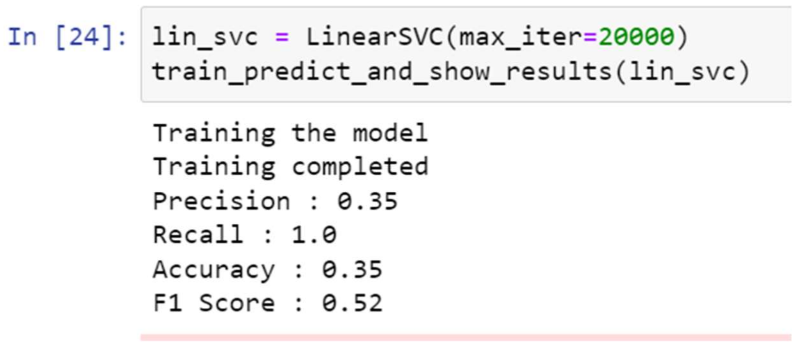
```
In [22]: log_reg = LogisticRegression()
         train_predict_and_show_results(log_reg)

         Training the model
         Training completed
         Precision : 1.0
         Recall : 0.0
         Accuracy : 0.65
         F1 Score : 0.0
```



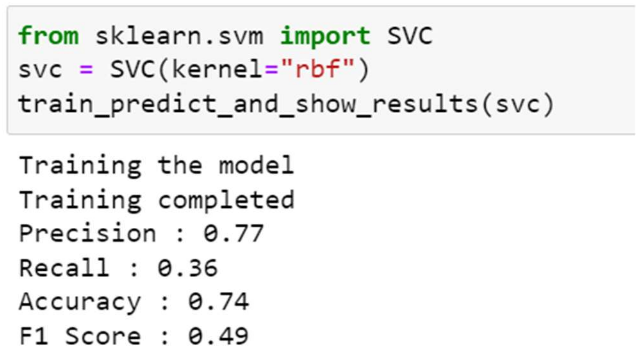
```
In [23]: forest = RandomForestClassifier()
         train_predict_and_show_results(forest)

         Training the model
         Training completed
         Precision : 0.81
         Recall : 0.93
         Accuracy : 0.9
         F1 Score : 0.87
```



```
In [24]: lin_svc = LinearSVC(max_iter=20000)
         train_predict_and_show_results(lin_svc)

         Training the model
         Training completed
         Precision : 0.35
         Recall : 1.0
         Accuracy : 0.35
         F1 Score : 0.52
```

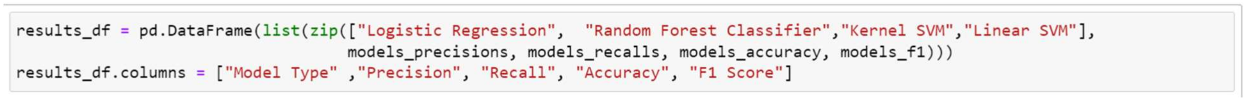


```
from sklearn.svm import SVC
svc = SVC(kernel="rbf")
train_predict_and_show_results(svc)

Training the model
Training completed
Precision : 0.77
Recall : 0.36
Accuracy : 0.74
F1 Score : 0.49
```

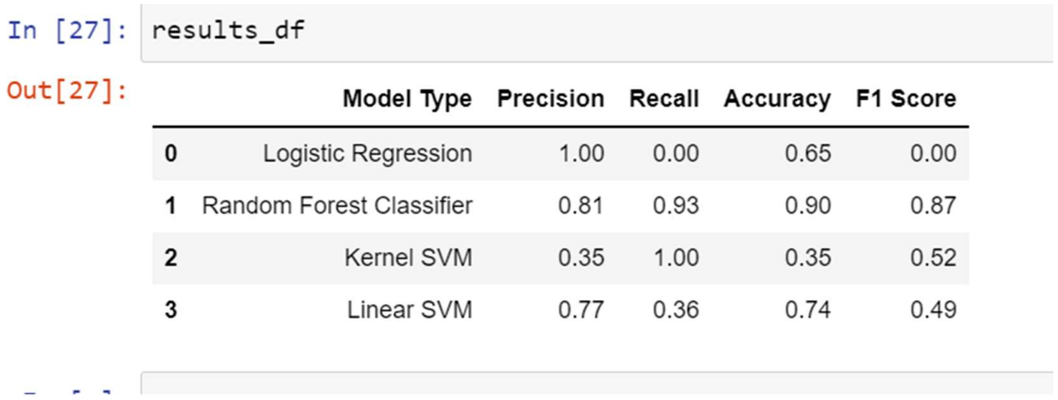
Conclusion:

Form the above features, Random Forest classifier has best accuracy of all the models better recall score and Precision and F1 score using the below code



```
results_df = pd.DataFrame(list(zip(["Logistic Regression",  "Random Forest Classifier","Kernel SVM","Linear SVM"],
                                   models_precisions, models_recalls, models_accuracy, models_f1)))
results_df.columns = ["Model Type" ,"Precision", "Recall", "Accuracy", "F1 Score"]
```

Forming a table using the models and accuracy, f1 scores.



```
In [27]: results_df
```

Out[27]:

	Model Type	Precision	Recall	Accuracy	F1 Score
0	Logistic Regression	1.00	0.00	0.65	0.00
1	Random Forest Classifier	0.81	0.93	0.90	0.87
2	Kernel SVM	0.35	1.00	0.35	0.52
3	Linear SVM	0.77	0.36	0.74	0.49