Inspiring Excellence

CSE422

Artificial Intelligence

Project Report On:

Pokémon Final Evolution Form Prediction

Group No: 08, Section-08

Prepared By:

Nisharga Nirjan (20101020)

Shaira Chowdhury (20101261)

Sultana Marium Trina (20101059)

30th April, 2023

Contents

Introduction:

Pokémon games are regarded as one of the most successful video game adaptations of all time and go way back to when the idea of video games were still afresh. Nintendo, on the other hand, is a major player in the video game industry and has a large team of developers working on creating new games for their consoles for decades. Therefore, working with this dataset can provide insights and opportunities for research, game development, and other renown fields.

The core objective of our project is to carefully observe specific input features namely ‘Base hit points stats’, ‘Base attack stats’, ‘Base defense stats’ etc. of each pokémon and predict their final evolution. In this case, the final evolution would work as our output label which holds binary values referring to ‘yes’ or ‘no’. Since the output label contains categorical variables, this would stand as a classification problem. By using classification models such as K-Nearest Neighbor (KNN), Decision Tree and Support Vector Machine (SVM), we plan on recovering the best model and drawing a brief analysis on the rest of the models. For the ease of understanding, we would also demonstrate our findings through heatmaps, bar charts and other visual stances.

Dataset Description:

Our project uses the “All Pokemon Dataset” that contains all the information for 1032 instances of different pokemons, with a total of **44 features** in the dataset such as: ‘Name’, ‘HP’, ‘Mean’, ‘Generation’, etc. Out of the 44 features, 5 of them are categorical features, namely: 'Name', 'Type 1', 'Type 2', 'Abilities', ‘Experience type’ and the rest 39 features are quantitative. So we have a total of **1032x44 data points**.

Our Project is **a classification problem** since the solution we’re trying to predict is whether a pokémon is in its final evolution form or not. As our output feature ‘Final Evolution’ has two unique classes, 1 and 0, this means that our project is trying to predict whether a pokémon is a legendary pokémon, so 1, or not, so 0. So, this indicates that our proposed model is attempting to classify the problem.

For our particular problem we only need to select a few features, we will explain this further in the data pre-processing section of this report. So a correlation between those particular features when visualized using a heatmap looks as such:

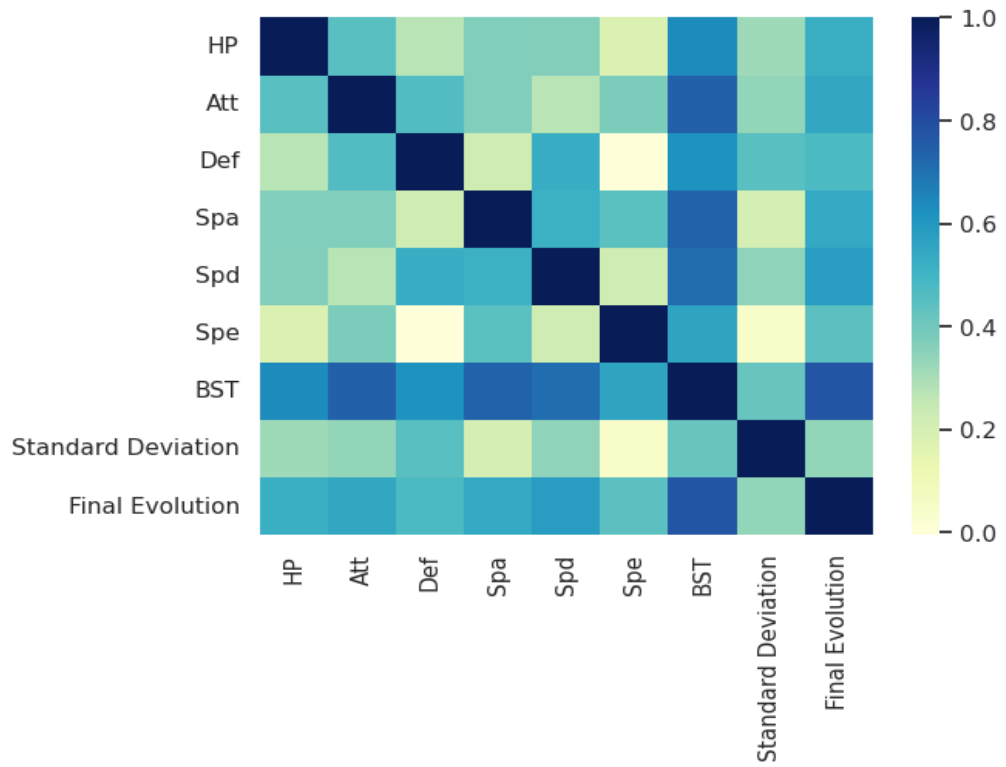

Fig: Heatmap of input and output features

As we can see from the above figure, when we measure the correlation between two features using the color gradient shown on the right, no certain feature is highly correlated with more than one feature as there is no column where the correlation value is greater than 0.75 at least twice. Thus all these features are necessary for our model to run properly.

To further explain our output feature, 'Final Evolution' as mentioned above has only two unique classes that are fairly balanced in the "All Pokemon Dataset". With 449 instances having the class value "0", meaning not in final evolution form, and the rest of the 583 instances being "1", in final evolution form. This somewhat reduced the chances of our model being biased. As shown in the bar chart:

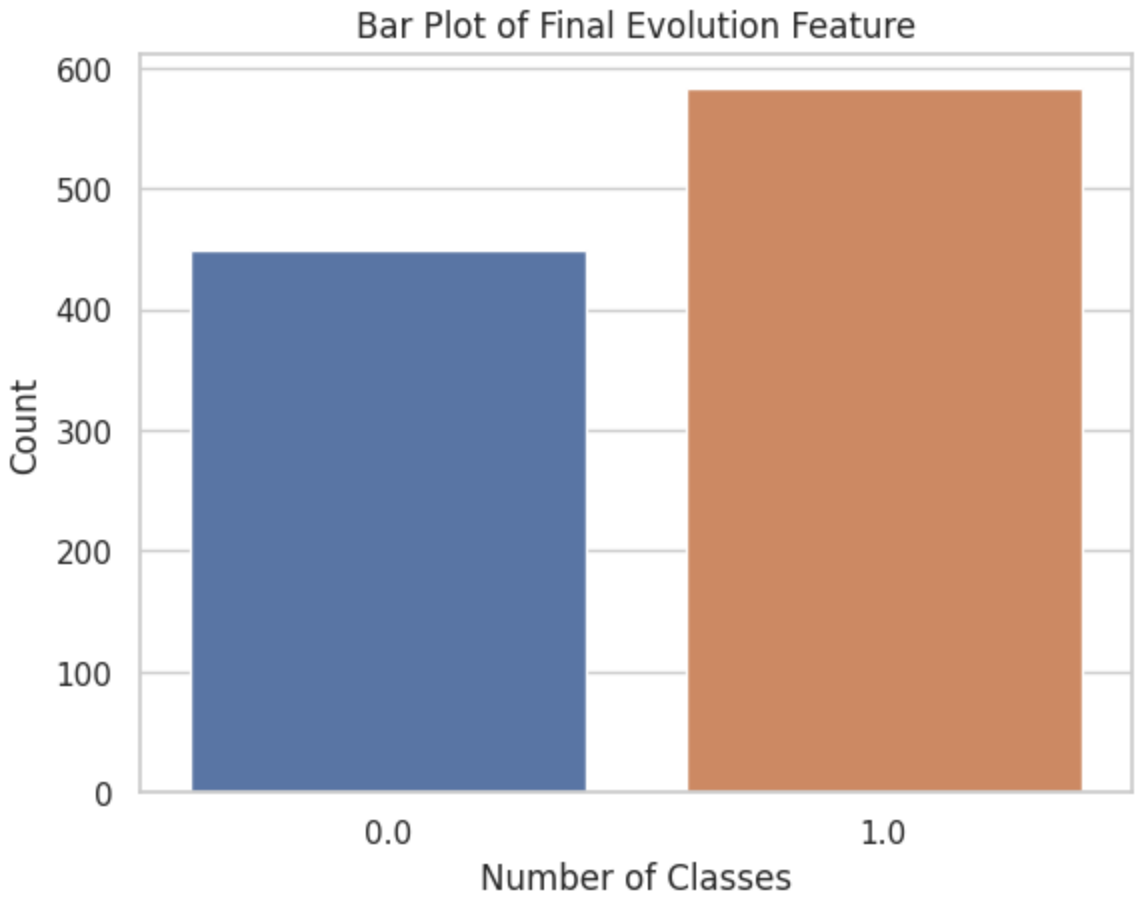

Fig: Bar chart of unique classes in the output feature

Source of the dataset:

Hernández, M. (2021). All Pokemon dataset. Kaggle,
https://www.kaggle.com/datasets/maca11/all-pokemon-dataset

Dataset Pre-Processing:

Faults and Solutions:

Categorical Values:

Out of the 44 features in our dataset 5 features represent categorical values, which are Name, Type 1, Type 2, Abilities and Experience type.

As these five categorical features are not required for our particular machine learning project and they did not contribute to the solution of the target problem, so instead of encoding we simply removed the features: Name, Type 1, Type 2, Abilities and Experience type.

Unnecessary Features:

After solving the 1st fault, out of the 39 features, 30 features did not contribute to the solution this model is trying to predict. As we have found that Pokémons in final evolution form have high HP, Att, Def, Spa, Spd, Spe, BST and Standard Deviation which are the base stats of the pokemon. Thus there is a pattern here for the model to learn. As the other features do not contribute to this pattern and will only reduce the performance of the model so our solution is to remove those unnecessary features.

Thus we have removed 30 unnecessary features which are: Number, Mean, Generation, Experience to level 100, Catch Rate, Legendary, Mega Evolution, Alolan Form, Galarian Form, Against Normal, Against Fire, Against Water, Against Electric, Against Grass, Against Ice, Against Fighting, Against Poison, Against Ground, Against Flying, Against Psychic, Against Bug, Against Rock, Against Ghost, Against Dragon, Against Dark, Against Steel, Against Fairy, Height, Weight and BMI.

Null Values:

After removing categorical and unnecessary features there are no null values in the dataset. Thus we don't have to remove any rows.

High Correlation Feature:

Of the remaining 9 features as seen from the correlation heatmap above, there are no features that have high correlation, of greater than 0.75, at least twice, thus we will not need to remove any features due to high correlation.

Dimensions:

Before pre-processing, dimension of dataset: **(1032, 44)**

After pre-proccesing, dimension of dataset: **(1032, 9)**

Dataset Splitting:

Among the 9 features 8 are our input features and 1, “Final Evolution”, is our output feature. So to split this dataset we have used two methods, **Random** and **Stratified**. However, in both these splitting methods we have kept the training and testing set to a **70%** and **30%** ratio respectively. With x_train_1 and y_train_1 being the training input and output set respectively with 70% of the data splitted using Random method, x_test_1 and y_test_1 being the text set containing 30% of data using Random method. On the other hand, x_train_2, y_train_2, x_test_2 and y_test_2 are the training and testing set split using the Stratified method into a 70-30 ratio.

Feature Scaling:

For our project we used two types of scalers in order to determine the best model later on in the comparison section. The two scalers used are MinMax and Standard Scalers. As we have two sets of data from splitting, random and stratified, so using two scalers gives us **4 sets** of training and testing sets of data for our model. We will pick the best result from these four in the comparison section.

Model Training and Testing:

Models Used:

1. **KNN - (K-Nearest Neighbors) :**

   KNN is a non-linear classifier and we use it on 4 different training and testing sets of data obtained from the pre-processing stage. By doing so we have four different models namely: KNN-MinMax-Random, KNN-Standard_Random, KNN-MinMax-Stratified and KNN-Standard-Stratified. We have calculated accuracy, precision, recall and F1 score for all 4 and received a fairly good score with all 4 score results for all 4 models being above 0.9.

2. **Decision Tree:**

   Decision Tree is a supervised learning algorithm which has a tree-like structure and can be used for classification problems. Again we have used this model on 4 different training and testing sets to get: Decision Tree-MinMax-Random, Decision Tree-Standard-Random, Decision Tree-MinMax-Stratified and Decision Tree-Standard-Stratified. Similarly with the decision tree we calculated all four types of scores for all four types of the model to get all the scores to be above 0.8. We have also created a visualization for the tree for better understanding -

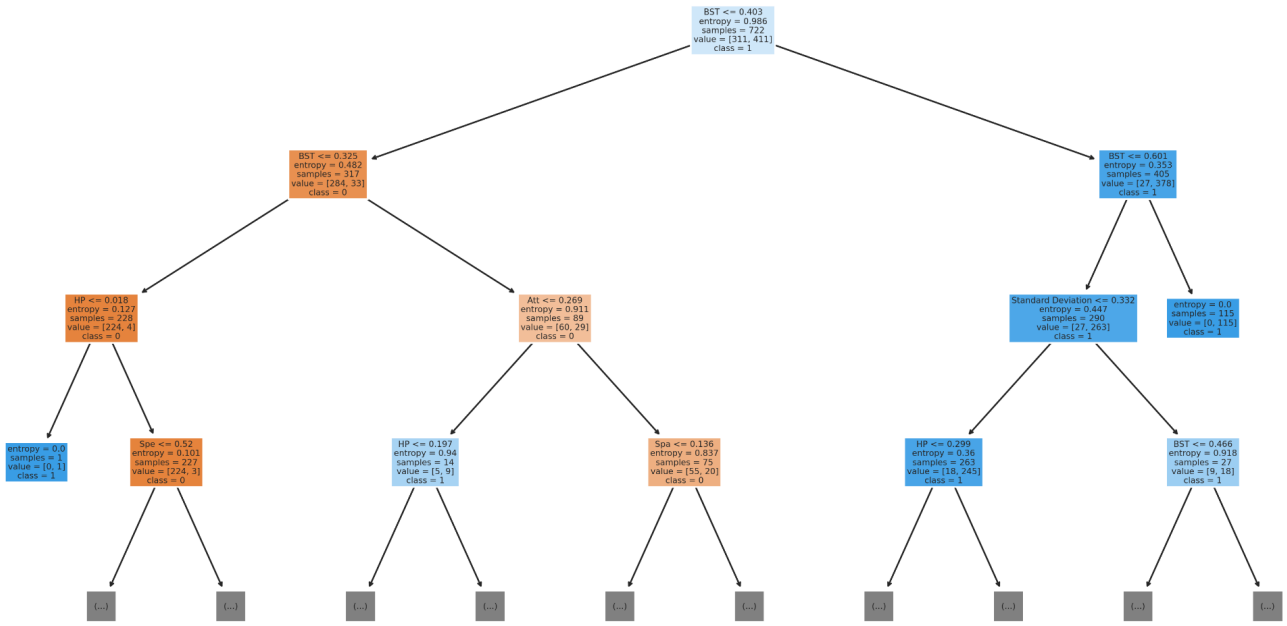

Fig: Decision Tree-MinMax-Random

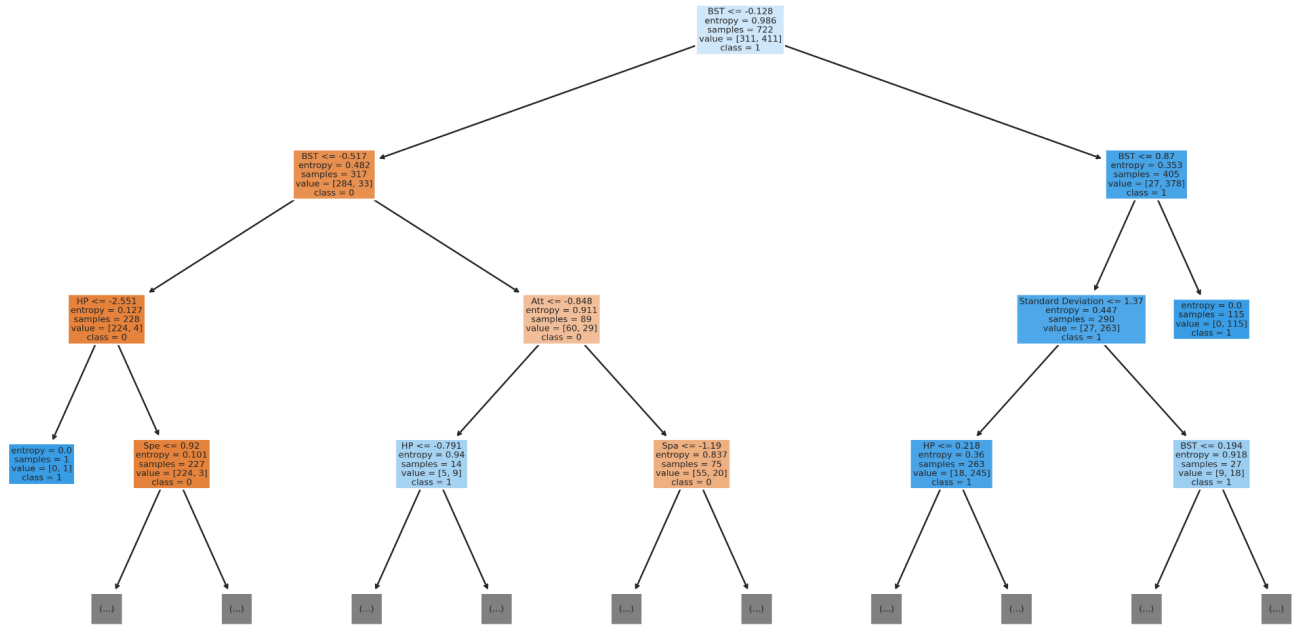

Fig: Decision Tree-Standard-Random

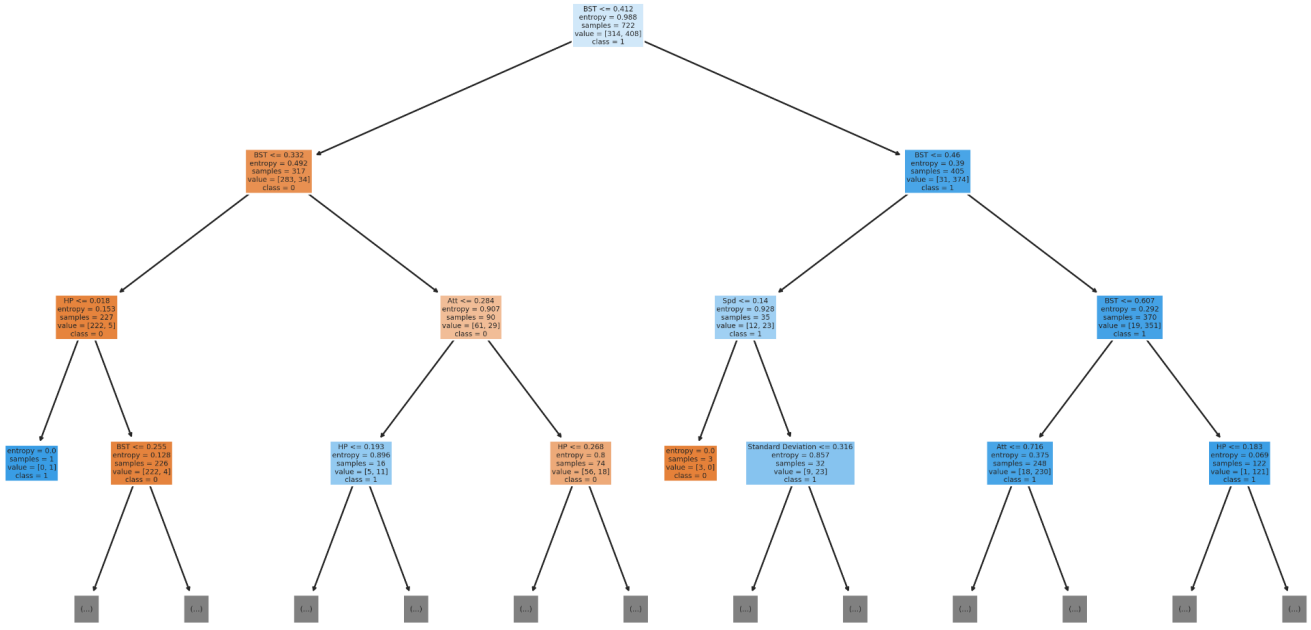

Fig: Decision Tree-MinMax-Stratified

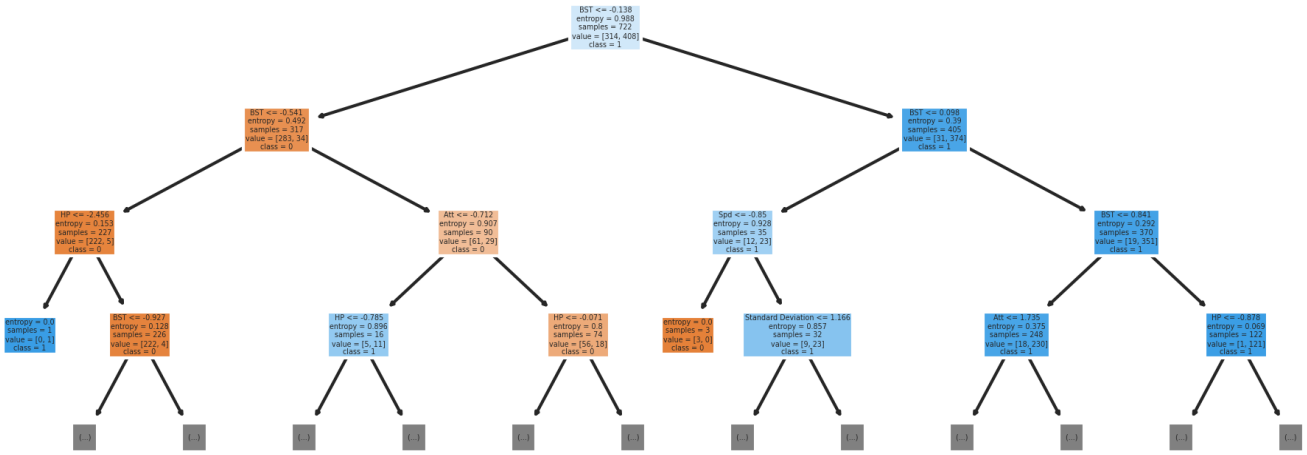

Fig: Decision Tree-Standard-Stratified

3. **SVM - (Support Vector Machine):**

   SVM can be used for classification problems which we used on all 4 of our training and testing sets to get: SVM-MinMax-Random, SVM-Standard-Random, SVM-MinMax-Stratified and SVM-Standard-Stratified. After calculating all four types of performance scores for all four types of the model to again get all the scores to be above 0.9.

Model Comparison and Selection Analysis:

With using 3 models on 4 types of training and testing sets we got 12 models to compare and analyze in our project.

The **Model Accuracy Score** of all these 12 models is represented in the bar chart below:

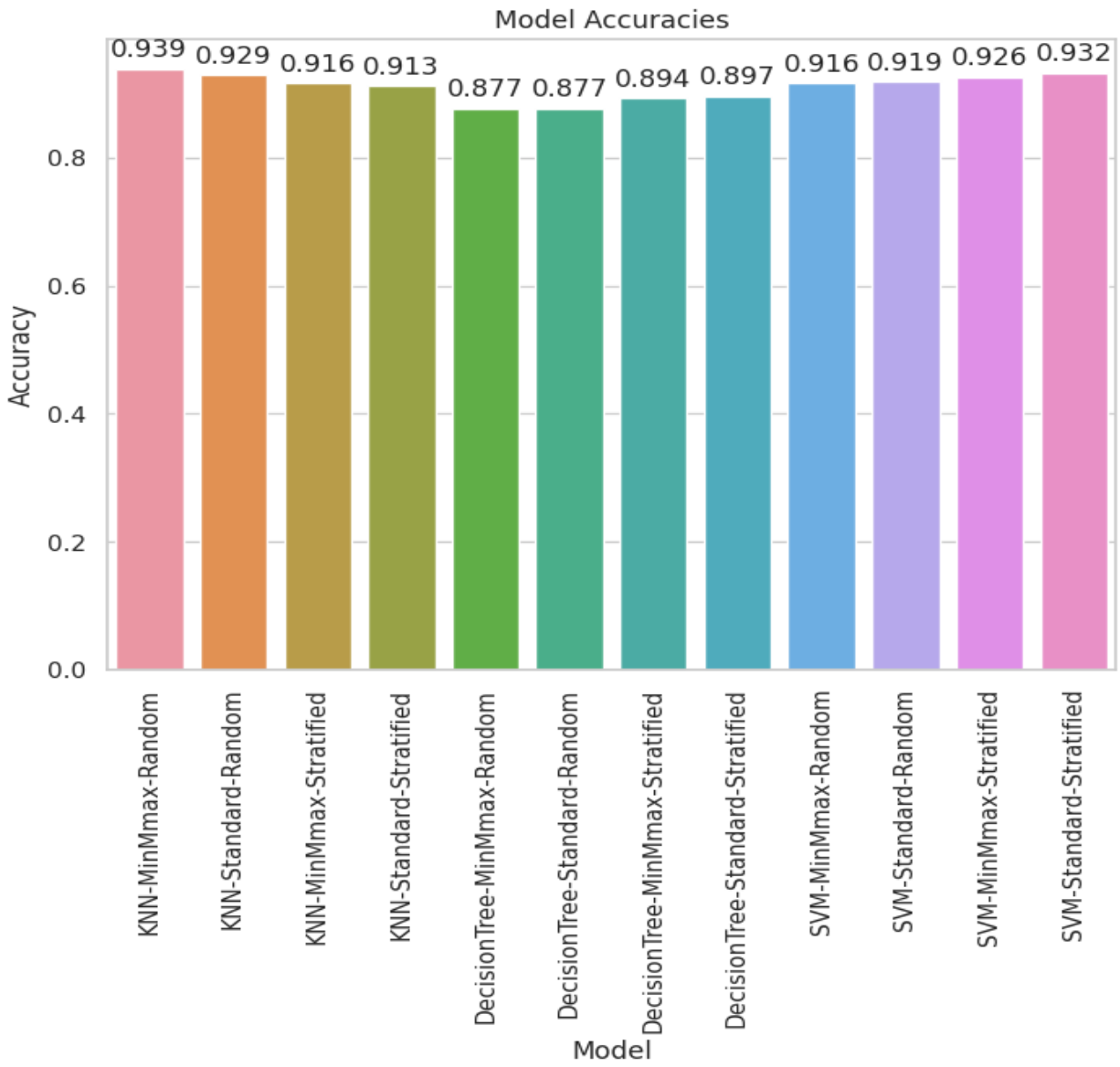

Fig: Accuracy Scores of 12 Classification Models

As we can see all the 12 models have given very high accuracy results and because accuracy alone is not a good performance measure, thus we need to analyze the other 3 scores, precision, recall and F1 score, to find the best performing model among the 12.

The **Model Precision Score** of all these 12 models is represented in the bar chart below:

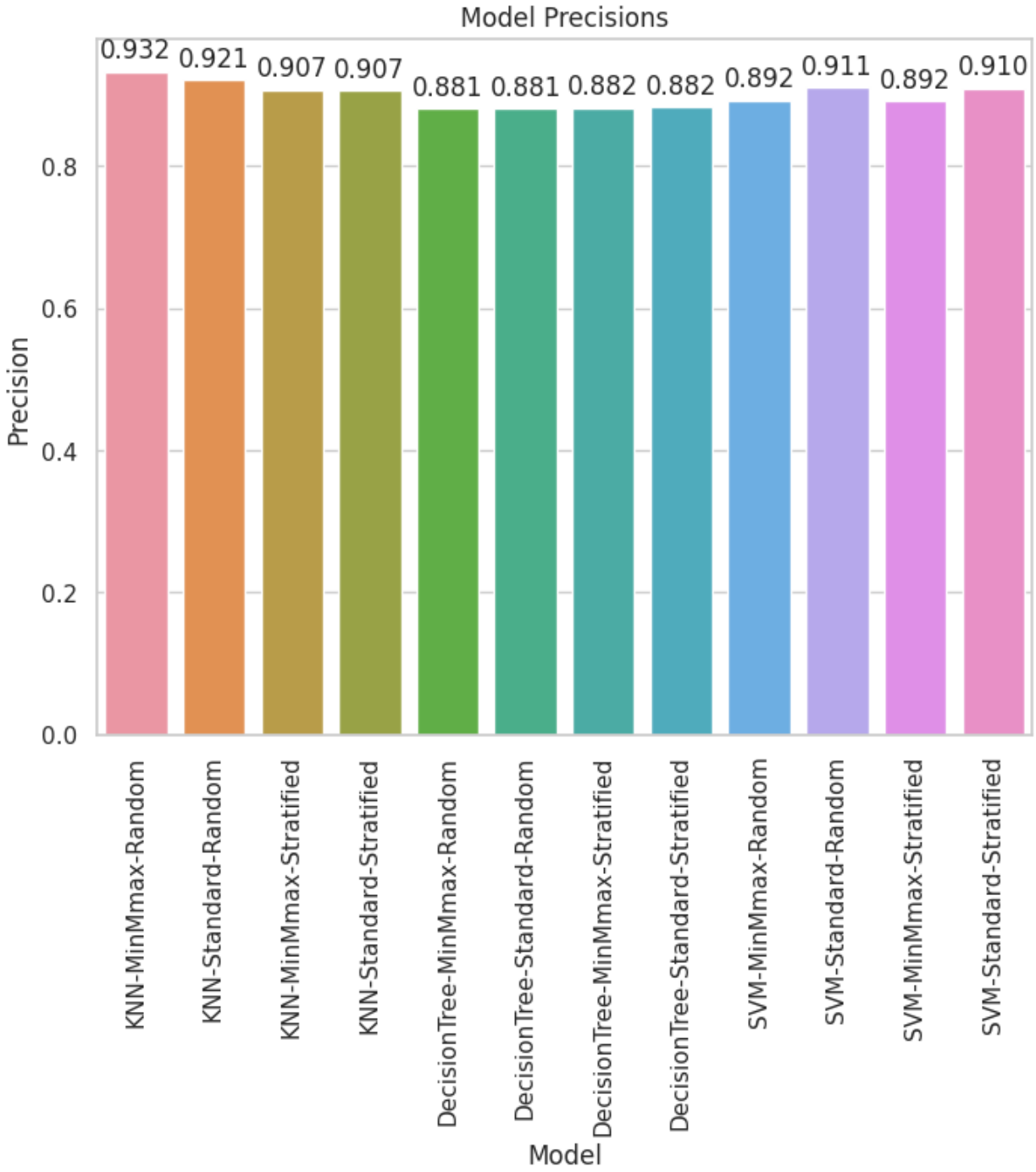

Fig: Precision Scores of 12 Classification Models

The **Model Recall Score** of all these 12 models is represented in the bar chart below:

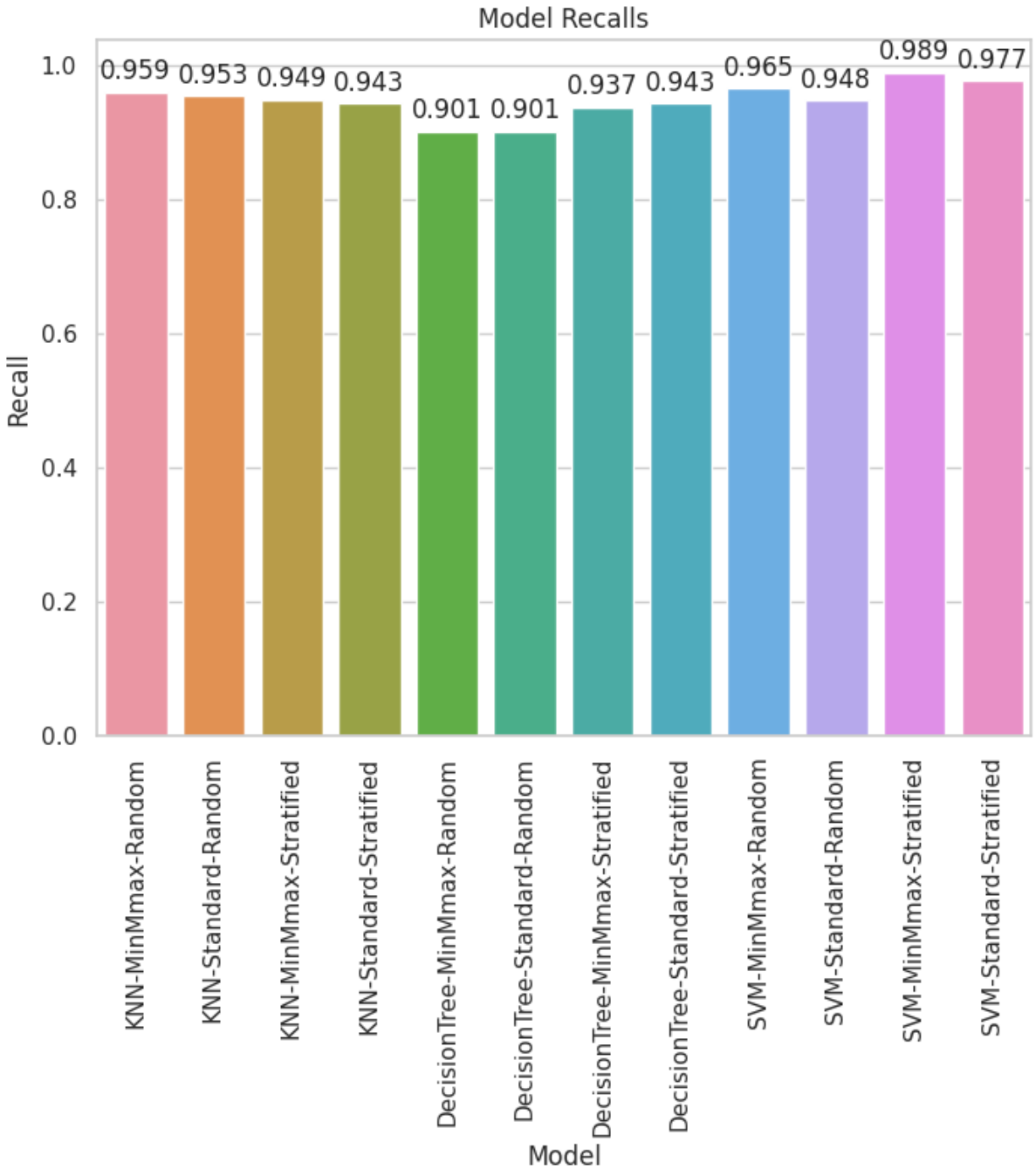

Fig: Recall Scores of 12 Classification Models

The **Model F1 Score** of all these 12 models is represented in the bar chart below:

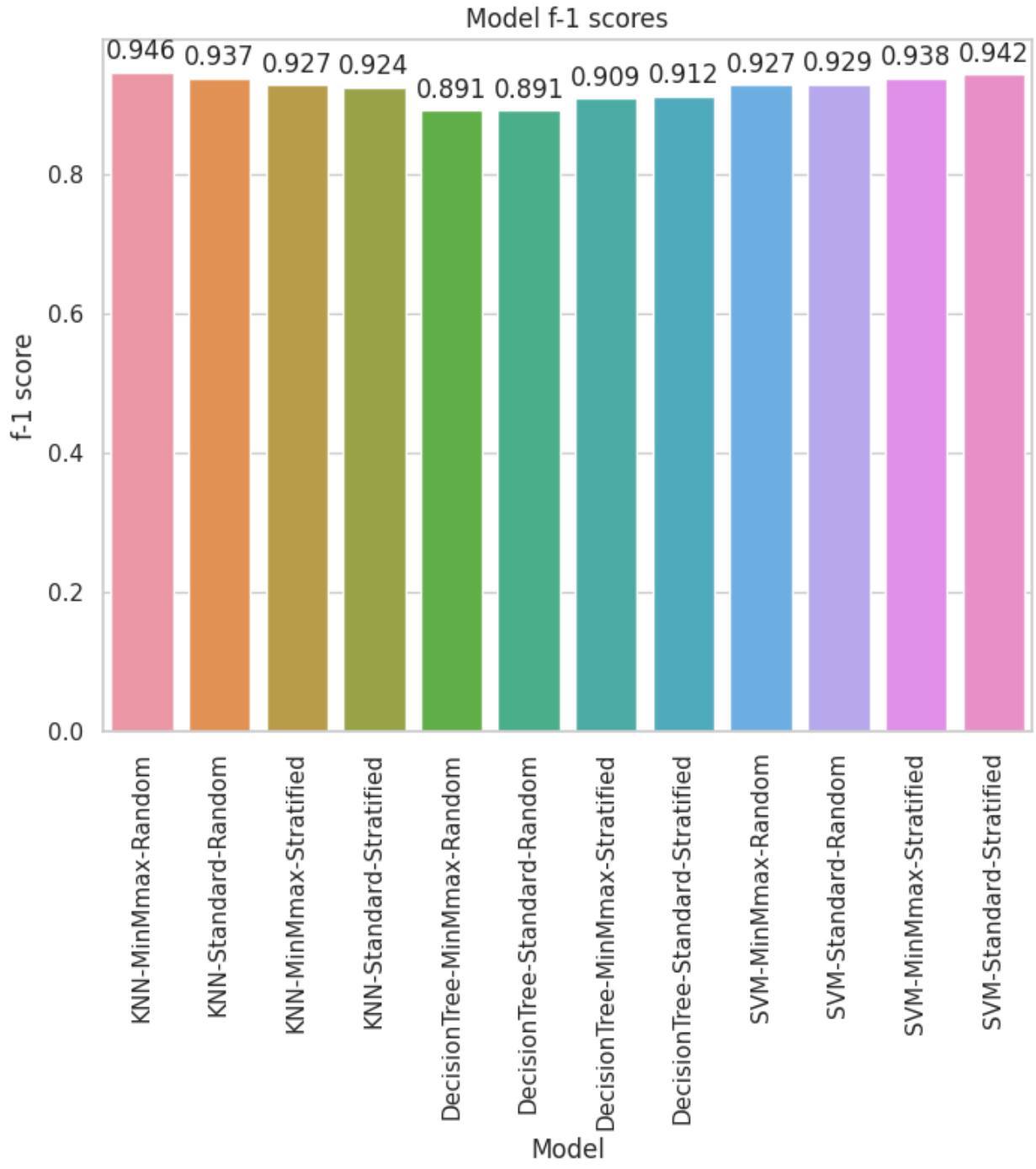

Fig: F1 Scores of 12 Classification Models

We also added a table representation of the scores to better identify the best **3** models from KNN, Decision Tree and SVM, 1 from each.

Models	Accuracy	Precision	Recall	F1 Score
KNN-MinMax-Random	**0.939**	**0.932**	**0.959**	**0.946**
KNN-Standard-Random	**0.929**	**0.921**	**0.953**	**0.937**
KNN-MinMax-Stratified	**0.916**	**0.907**	**0.949**	**0.927**
KNN-Standard-Stratified	**0.913**	**0.907**	**0.943**	**0.924**
Decision Tree-MinMax-Random	**0.877**	**0.881**	**0.901**	**0.891**
Decision Tree-Standard-Random	**0.877**	**0.881**	**0.901**	**0.891**
Decision Tree-MinMax-Stratified	**0.894**	**0.882**	**0.937**	**0.909**
Decision Tree-Standard-Stratified	**0.897**	**0.882**	**0.943**	**0.912**
SVM-MinMax-Random	**0.916**	**0.892**	**0.965**	**0.927**
SVM-Standard-Random	**0.919**	**0.911**	**0.948**	**0.929**
SVM-MinMax-Stratified	**0.926**	**0.892**	**0.989**	**0.938**
SVM-Standard-Stratified	**0.932**	**0.910**	**0.977**	**0.942**

Fig: Table representing accuracy, precision, recall and F-1 scores of all the models

From the table above, it can be clearly stated that the best model from among KNN is the **KNN-MinMax-Random** model which has a model accuracy value of 0.939, model precision value of 0.932, model recall value of 0.959 and an F-1 score of 0.946 which is higher than the other 3 models.

Similarly, from among the 4 available Decision Tree models, the **Decision Tree-Standard-Stratified** model is the best since it has a model accuracy value of 0.897, model precision value of 0.882, model recall value of 0.943 and an F-1 score of 0.912.

Lastly, the **SVM-Standard-Stratified** model can be appointed as the best among the 4 available SVM models as it has a model accuracy value of 0.932, model precision value of 0.910, model recall value of 0.977 and an F-1 score of 0.942. However it is important to note that even if the

precision and recall values available are slightly lower than other available models, we are prioritizing the F-1 score here, as it is a ratio of the precision and recall values.

Heatmap Representation of Confusion Matrix

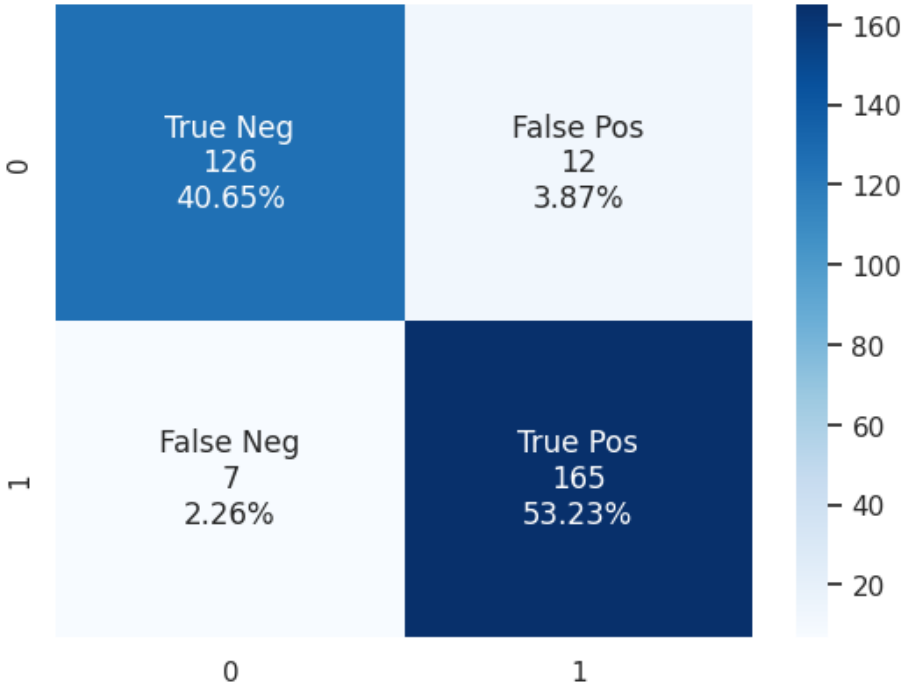

Fig: Heatmap representation of the confusion matrix for KNN-MinMax-Random

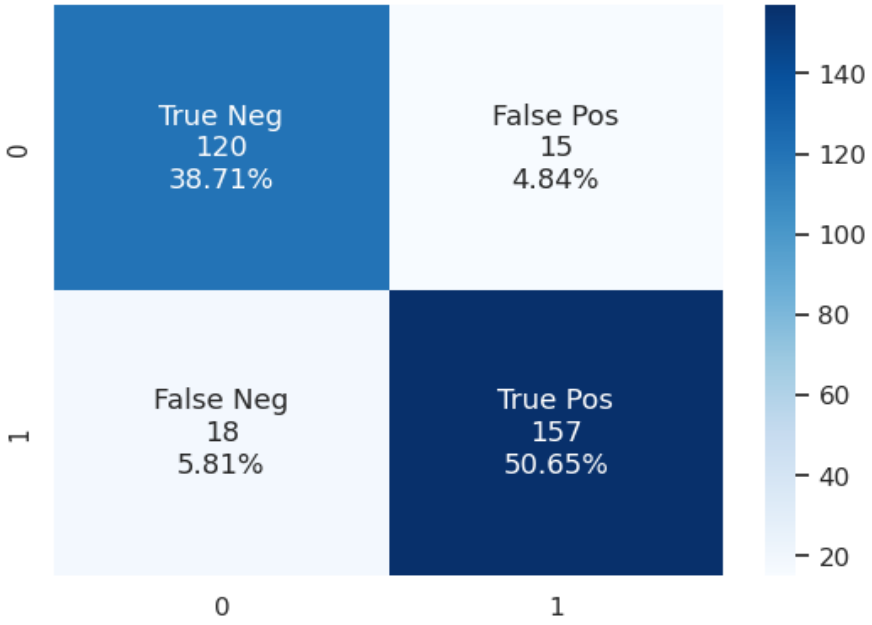

Fig: Heatmap representation of the confusion matrix for Decision Tree-Standard-Stratified

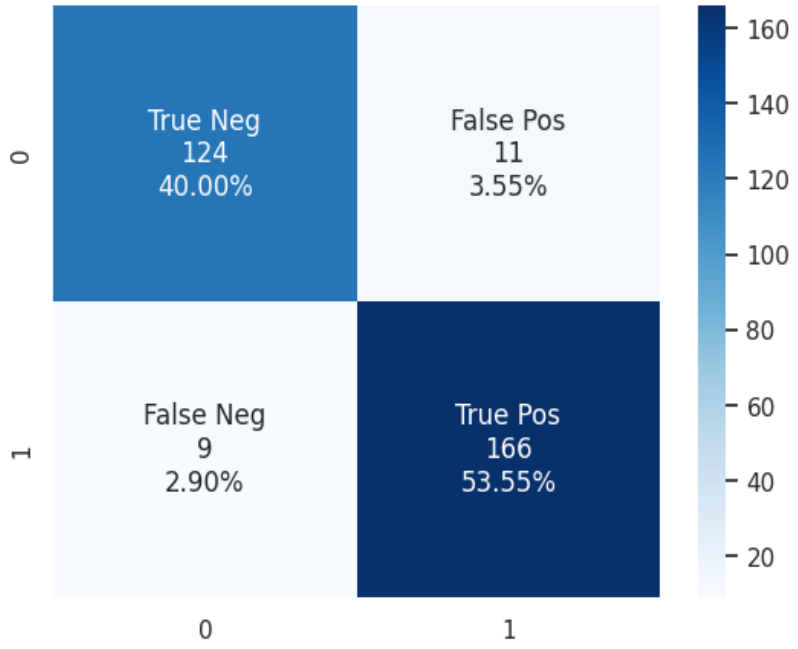

Fig: Heatmap representation of the confusion matrix for SVM-Standard-Stratified

Conclusion:

After splitting and scaling the data and feeding those to our 3 proposed models namely: K-Nearest Neighbor (KNN), Decision Tree and Support Vector Machine (SVM), we came to a conclusion that the **KNN-MinMax-Random model** is by far the **best** among them. The KNN-MinMax-Random model holds values above **0.93** in all 4 of its columns and easily surpasses the other available models. Although the recall value of SVM-Standard-Stratified model is slightly higher than that of KNN-MinMax-Random model, by looking at their corresponding heatmap representation of confusion matrix, which holds nearly **identical True Positive** values, it can safely be assumed that theKNN-MinMax-Random model is indeed the best model for our project.