SPOTIFY MUSIC GENRE PREDICTION

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELORS OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

By

GUNTUR SIRI CHANDANA

Regd. No.: 20B91A0488

Under Supervision of Mr. Gundala Nagaraju

Henotic Technology Pvt Ltd, Hyderabad

(Duration: 7th July, 2022 to 6th September, 2022)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(An Autonomous Institution)

Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada

CHINNA AMIRAM, BHIMAVARAM,

ANDHRA PRADESH

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(An Autonomous Institution)

CHINNA AMIRAM, BHIMAVARAM

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the "**Summer Internship Report"** submitted by **Miss. Guntur. Siri Chandana (20B91A0488)** is the work done by her and submitted during 2021 - 2022 academic year, in partial fulfillment of the requirements for the award of the Summer Internship Program for **Bachelors of Technology in Electronics and Communication** at **Henotic technology** from 05.07.2022 to 04.09.2022.

Department Internship Coordinator **Dean -T & P Cell** **Head of the Department**

Table of Contents

Abstract

Music is a part of human society and an essential part of cultural heritage. When artists create new music, many factors affect them. From these data, determining whether the characteristics mean a revolution in music development and capturing the parameters of “music influence” may become an important research topic. In this report, we use machine learning methods to mine influencers' influence on the main genres of followers, construct an influence prediction model for influencers, and capture the “music influence” parameters. The proposed model is evaluated on the music_genre.csv data set provided by spotify. We extract the information features in the data set and combine machine learning methods to mine influencers' influence on the main genres of followers. Compare multiclass classification algorithms to predict whether the type of follower is the same as the type of influencer. After a repeated iterative search on the parameters of each model using the grid search method, the prediction effects of the three machine learning models were compared.

1.0 Introduction

With the increasing power of computer technology, companies and institutions can nowadays store large amounts of data at reduced cost. The amount of available data is increasing exponentially and cheap disk storage makes it easy to store data that previously was thrown away. There is a huge amount of information locked up in databases that is potentially important but has not yet been explored. The growing size and complexity of the databases makes it hard to analyse the data manually, so it is important to have automated systems to support the process. Hence there is the need of computational tools able to treat these large amounts of data and extract valuable information.

In this context, Data Mining provides automated systems capable of process in garge amounts of data that are already present in databases. Data Mining is used to automatically extract important patterns and trends from databases seeking regularities or patterns that can reveal the structure of the data and answer business problems. Data Mining includes learning techniques that fall into the field of Machine learning. The growth of databases in recent years brings data mining at the forefront of new business technologies.

A key challenge for the insurance industry is to charge each customer an appropriate price for the risk they represent. Risk varies widely from customer to customer and a deep understanding of different risk factors helps predict the likelihood and cost of insurance claims. The goal of this program is to see how well various statistical methods perform in predicting auto Insurance claims based on the characteristics of the driver, vehicle and driver / vehicle coverage details.

A number of factors will determine BI claims prediction among them a driver's age, past accident history, and domicile, etc. However, this contest focused on the relationship between claims and vehicle characteristics well as other characteristics associated with the auto insurance policies.

1.1. What are the different types of Machine Learning?

Machine learning is a subset of AI, which enables the machine to automatically learn from data, improve performance from past experiences, and make predictions. Machine learning contains a set of algorithms that work on a huge amount of data. Data is fed to these algorithms to train them, and on the basis of training, they build the model & perform a specific task

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning:

As its name suggests, supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y)

Supervised machine learning can be classified into two types of problems, which are given below:

- Classification

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "**Yes" or No, Male or Female, Red or Blue, etc**. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are **Spam Detection, Email filtering, etc.**

- Regression

Regression algorithms are used to solve regression problems in which there is a **linear relationship between input and output variables**. These are used to predict continuous output variables, such as **market trends, weather prediction, etc.**

2. Unsupervised Machine Learning:

Unsupervised Learning is different from the supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabelled dataset, and the machine predicts the output without any supervision.

Unsupervised Learning can be further classified into two types, which are given below:

- Clustering

The clustering technique is used when we want to **find the inherent groups from the data**. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups. An example of the clustering algorithm is **grouping the customers by their purchasing behaviour**.

- Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in **Market Basket analysis, Web usage mining, continuous production**, etc.

3. Semi-Supervised Machine Learning:

Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning. It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabelled datasets during the training period.

4. Reinforcement Learning:

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance. Agent gets rewarded for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.

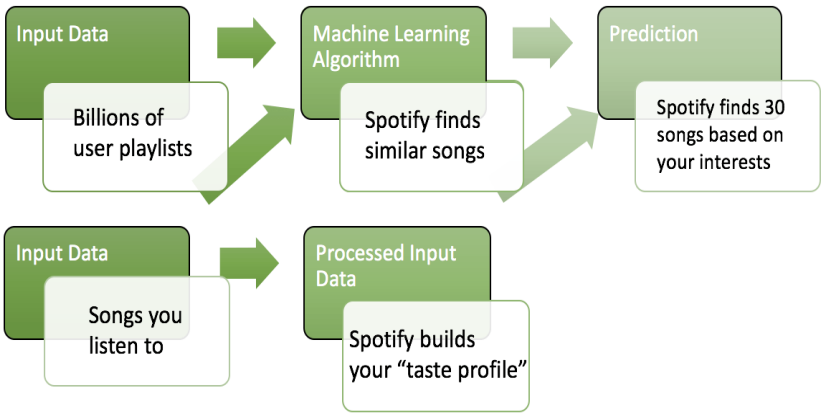
In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only.

1.2. Benefits of Using ML in Music Genre Prediction

- It is used to help listeners discover content via recommendations and search.
- Generate playlists.
- Separate Music according to respective genre.
- Create Mood Based Playlists
- Extract audio content-rich signals for cataloguing
- Create music with AI-assisted tools

Prediction of consumer lifetime value and segmentation of consumers are some of the significant challenges the advertisers face today. The business has exposure to vast amounts of data, which can be used easily to provide insightful insights into the Market. ML and data mining will help companies to forecast consumer habits, purchasing trends, and improve individual customers to submit best-possible deals based on their surfing and purchase experience.

ML can help to improve customer loyalty and also provide a better customer experience. It is done by using the previous call logs to evaluate the customer behaviour and relying on the accurate transfer of the client request to the most suitable customer service director. It dramatically reduces the cost and time spent on customer relationship management. For this reason, large organizations use predictive algorithms to provide suggestions to their customers about the products they enjoy.



1.3. About Spotify

Spotify transformed music listening forever when it launched in 2008. Discover, manage and share over 80 million tracks, including more than 4 million podcast titles, for free, or upgrade to Spotify Premium to access exclusive features for music including improved sound quality and an on-demand, offline, and ad-free music listening experience. Today, Spotify is the world's most popular audio streaming subscription service with 433M users, including 188M subscribers, across 183 markets.

1.3.1 AI / ML Role in Spotify

In terms of technology, it uses AI, Big data analyses, and Machine learning to upgrade and customize the music expertise for the listener. There is no doubt that Spotify is one of the best music streaming apps on the market.

Spotify is best known for its excellent user interface and continuously improving music recommendations. Every Monday, each consumer receives a brand new playlist of new advocated songs designed for their own personalized choice predicated in their own listening history and also the songs they're most enthusiastic about.

Such exceptional use of machine learning to deliver the best service possible to its customers is an evolutionary step in future technology.

The reason is its extensive use of available resources and innovative future technologies like machine learning, AI, Big Data to assist business processes.

Spotify has revolutionized the music app industry and showed the world how to make the best use of the technology.

2.0 Spotify Music Genre Prediction

Auto insurance premiums have historically been priced on underwriting and rating. Underwriting is a process where the insurer assesses the applicant's risk. They do this by incorporating personal information and internal claims data into weighted algorithms. Insurers then look at rating factors to predict the likelihood of a claim's submission. The rating assigns a price based on the projected cost to the insurer of assuming financial responsibility of potential claims. Auto insurance premiums fluctuate with the projected risk to the insurer. A policyholder can lower their premium by taking on more risk. For instance, the policyholder can choose to drop optional coverages or increase the deductible. A deductible is the out-of-pocket portion of the claim for which the driver is responsible.

The main factors for auto insurance BI claims are Location, Age, Gender, Marital status, driving experience, driving record, Claims history, Credit history, Previous insurance coverage, Vehicle type, Vehicle use, Miles driven, Coverages and deductibles.

2.1. Main Drivers for AI Spotify Music Genre Prediction

Predictive modelling allows for simultaneous consideration of many variables and quantification of their overall effect. When a large number of claims are analysed, patterns regarding the characteristics of it that drive loss development begin to emerge.

The following are the main drivers which influencing the Predictive Analytics:

• Popularity • Acousticness • Danceability • Energy • Instrumentalness • Key • Liveness	• Loudness • Mode • Speechiness • Tempo • Valence

2.2. Internship Project - Data Link

The internship project data has taken from Kaggle and the link is Music genre | Kaggle

3.0 AI / ML Modelling and Results

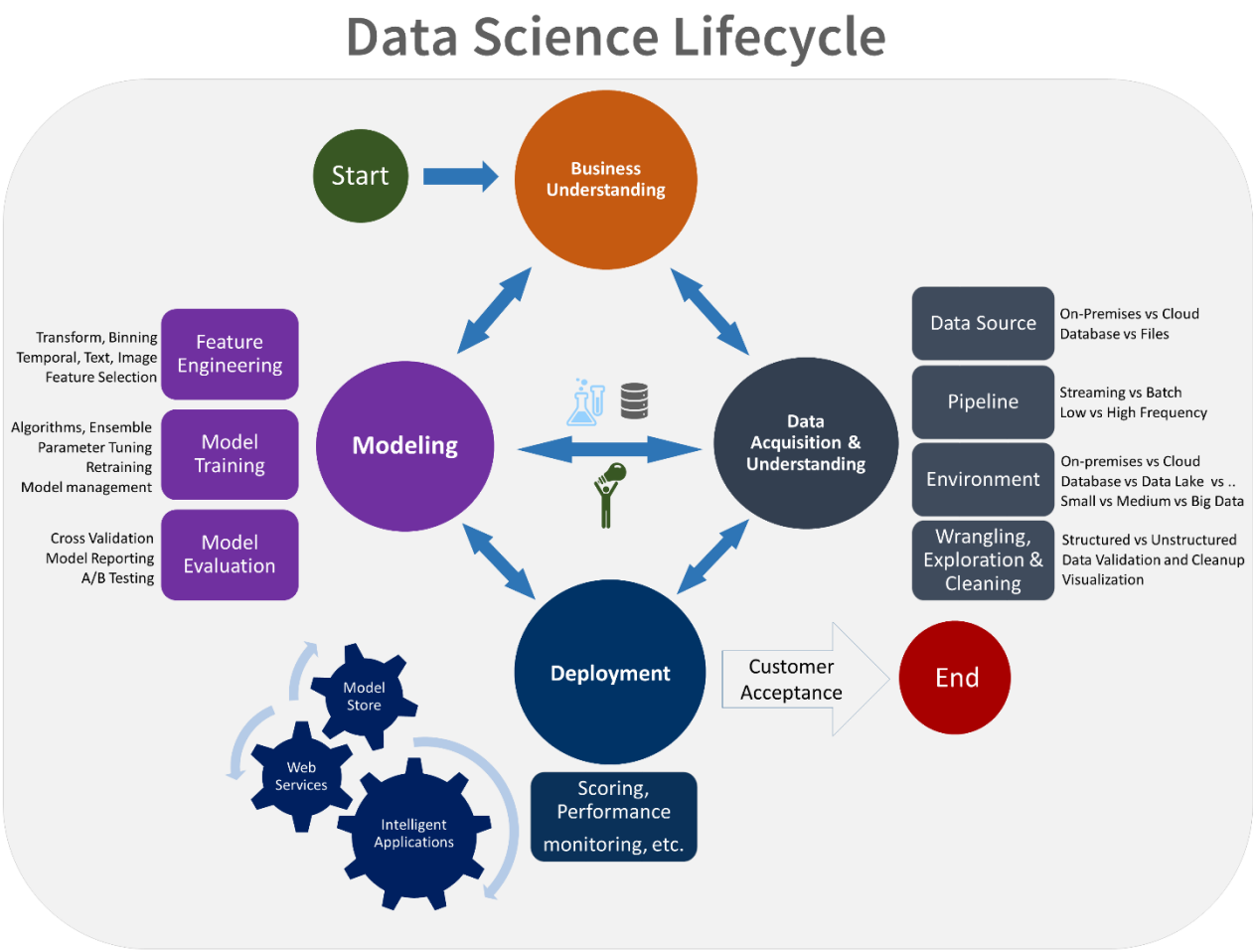
3.1. Spotify Music Genre Prediction

Predictive models are most effective when they are constructed using a company's own historical data since this allows the model to recognize the specific nature of a company's exposure as well as its practices. The construction of the model also involves input from the company throughout the process, as well as consideration of industry leading claims practices and benchmarks.

Predictive modelling can be used to quantify the impact to the claims department resulting from the failure to meet or exceed claim service leading practices. Proper use of predictive modelling will allow for potential savings across two dimensions:

3.2. Data Science Project Life Cycle

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data.



3.2.1 Data Exploratory Analysis

Exploratory data analysis has been done on the data to look for relationship and correlation between different variables and to understand how they impact or target variable.

3.2.2 Data Pre-processing

We removed variables which does not affect our target variable (Claimed Target) as they may add noise and also increase our computation time, we checked the data for anomalous data points and outliers. We did principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have greater correlation with our target variable.

3.2.2.1. Check the Duplicate and low variation data

Duplicates are checked to see whether the data set have any similar rows in it

To find duplicates on a specific column, we can simply call duplicated ( ) method on the column. The results are a Boolean series with value denoting the duplicate. In other words, the value true means the entry is identical to a previous one.

In the given data set there are 4 duplicated values so we can delete it or fill them if they're nulls

3.2.2.2. Identify and address the missing variables

For various reasons, many real world datasets contain missing values, often encoded as blanks, NaNs or other placeholders. Such datasets however are incompatible with scikit-learn estimators which assume that all values in an array are numerical, and that all have and hold meaning. A basic strategy to use incomplete datasets is to discard entire rows and/or columns containing missing values. However, this comes at the price of losing data which may be valuable (even though incomplete). A better strategy is to impute the missing values, i.e., to infer them from the known part of the data

- **Simple Imputer**

  The Simple Imputer class provides basic strategies for imputing missing values. Missing values can be imputed with a provided constant value, or using the statistics (mean, median or most frequent) of each column in which the missing values are located. This class also allows for different missing values encodings.

- **KNN Imputer**

  The **KNN Imputer** class provides imputation for filling in missing values using the k-Nearest Neighbours approach. By default, a Euclidean distance metric that supports missing values, nan Euclidean distances, is used to find the nearest neighbours. Each missing feature is imputed using values from n neighbours nearest neighbours that have a value for the feature. The feature of the neighbours are averaged uniformly or weighted by distance to each neighbour.

3.2.2.3. Handling of Outliers

Outliers are **extreme values that deviate from other observations on data** , they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample.

Here I haven't checked for outliers

3.2.2.4. Categorical data and Encoding Techniques

The performance of a machine learning model not only depends on the model and the hyper parameters but also on how we process and feed different types of variables to the model. Since most machine learning models only accept numerical variables, pre-processing the categorical variables becomes a necessary step. We need to convert these categorical variables to numbers such that the model is able to understand and extract valuable information.

Different encoding techniques:

- Label encoding or Ordinal encoding
- One hot encoding
- Dummy encoding
- Effect encoding
- Binary encoding
- Base N encoding
- Hash encoding
- Target encoding

3.2.2.5. Feature Scaling

In Data Processing we try to change the data in such a way that the model can process it without any problems. Feature Scaling is one such process in which we transform the data into a better version. Feature scaling is done to normalize the features in the dataset into a finite range.

There are several ways to do feature scaling .The below scaling methods are the top5 of the most commonly used feature scaling techniques.

- Absolute Maximum Scaling
- Min-Max Scaling
- Normalization
- Standardization
- Robust Scaling

3.2.3 Selection of Dependent and Independent variables

The dependent or target variable here is Claimed Target which tells us a particular policy holder has filed a claim or not the target variable is selected based on our business problem and what we are trying to predict.

The independent variables are selected after doing exploratory data analysis and we used Boruta to select which variables are most affecting our target variable.

3.2.4 Data Sampling Methods

The data we have is highly unbalanced data so we used some sampling methods which are used to balance the target variable so we our model will be developed with good accuracy and precision. We used three Sampling methods

3.2.4.1. Stratified sampling

Stratified sampling randomly selects data points from majority class so they will be equal to the data points in the minority class. So, after the sampling both the class will have same no of observations.

It can be performed using strata function from the library sampling.

3.2.4.2. Simple random sampling

Simple random sampling is a sampling technique where a set percentage of the data is selected randomly. It is generally done to reduce bias in the dataset which can occur if data is selected manually without randomizing the dataset.

We used this method to split the dataset into train dataset which contains 70% of the total data and test dataset with the remaining 30% of the data.

3.2.5 Models Used for Development

We built our predictive models by using the following ten algorithms

3.2.5.1. Model 01: Logistic Regression

Logistic uses logit link function to convert the likelihood values to probabilities so we can get a good estimate on the probability of a particular observation to be positive class or negative class. The also gives us p-value of the variables which tells us about significance of each independent variable.

3.2.5.2. Model 02: Decision Tree Classifier

Decision Tree is a supervised learning technique that can be used for both classification and regression problems, but mostly it is preferred for solving classification problems. It is a tree-structured algorithm, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

The decisions or the test are performed on the basic of features of the given datase

3.2.5.3. Model 03: Random Forest Classifier

Random forest is an algorithm that consists of many decision trees. It was first developed by Leo Breiman and Adele Cutler. The idea behind it is to build several trees, to have the instance classified by each tree, and to give a "vote" at each class. The model uses a "bagging" approach and the random selection of features to build a collection of decision trees with controlled variance. The instance's class is to the class with the highest number of votes, the class that occurs the most within the leaf in which the instance is placed.

The error of the forest depends on:

- Trees correlation: the higher the correlation, the higher the forest error rate.

- The strength of each tree in the forest. A strong tree is a tree with low error. By using trees that classify the instances with low error the error rate of the forest decreases.

3.2.5.4. Model 04: Extra Tree Classifier

This class implements a meta estimator that fits a number of randomized decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting

3.2.5.5. Model 5: KNN Classifier

K-Nearest Neighbor is one of the simplest machine learning algorithms based on supervised Learning technique. It uses proximity to make classifications or predictions about the grouping of an individual data point.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K-NN algorithm.

K-NN can be used for both classification and regressions but mostly it is used for the classification problems

3.2.5.6. Model 6: Support Vector Machine

Support Vector Machine is one of the most popular Supervised Learning algorithms, which is used for classifications well as Regression problems. However, it is primarily used for classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future .This best decision boundary is called a hyperplane

3.2.5.7. Model 7: Bagging Classifier

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples (or data) from the original training dataset

where N is the size of the original training set. Training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

3.2.5.8. Model 8: Gradient Boosting Classifier

This algorithm builds an additive model in a forward stage-wise fashion it allows for the optimization of arbitrary differentiable loss functions. In each stage n_classes_regression trees are fit on the negative gradient of the loss function, e.g., binary or multiclass log loss. Binary classification is a special case where only a single regression tree is induced.

3.2.5.9. Model 9: Light GBM

Light GBM is a gradient boosting framework based on decision trees to increase the efficiency of the model and reduces the memory usage.

It uses two novel techniques; Gradient based One side sampling and exclusive feature Bunding (EFB) which fulfills the limitations of histogram- based algorithm that is primarily used in all GBDT frameworks .The two techniques of GOSS and EFB describes form the characteristics of light GBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks.

3.2.5.10. Model 10: Gaussian Naïve Bayes

Naïve Bayes is a probabilistic machine learning algorithm used for many classification functions and is based on the Bayes Theorem.

Gaussian Naïve Bayes is the extension of the naïve bayes. While other functions are used to estimate data distribution, Gaussian or normal distribution is the simplest to implement as you will need to calculate the mean and standard deviation for the training data

3.3. AI / ML Models Analysis and Final Results

We used our train dataset to build the above models and used our test data to check the accuracy and performance of our models.

We used confusion matrix to check accuracy, Precision, Recall and F1 score of our models and compare and select the best model for given auto dataset of size ~ 50005

3.3.1 Different Model codes:

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier

from sklearn.ensemble import GradientBoostingClassifier
import lightgbm as lgb



# Create objects of classification algorithm with default hyper-parameters
```

```
ModelLR = LogisticRegression()
ModelDC = DecisionTreeClassifier()
ModelRF = RandomForestClassifier()
ModelET = ExtraTreesClassifier()
ModelKNN = KNeighborsClassifier(n_neighbors=5)
ModelSVM = SVC(probability=True)

modelBAG = BaggingClassifier(base_estimator=None, n_estimators=100, max_samples=1.0, max_features=1.0,
                             bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False,
                             n_jobs=None, random_state=None, verbose=0)

ModelGB = GradientBoostingClassifier()
ModelLGB = lgb.LGBMClassifier()
ModelGNB = GaussianNB()

# Evalution matrix for all the algorithms


MM = [ModelLR, ModelDC, ModelRF, ModelET, ModelKNN, ModelSVM, modelBAG, ModelGB, ModelLGB, ModelGNB]
counter=1
for models in MM:
    overall_tp=0
    overall_fn=0
    overall_fp=0
    overall_tn=0
    # Train the model training dataset

    models.fit(x_train, y_train)

    # Prediction the model with test dataset

    y_pred = models.predict(x_test)
```

```
    y_pred_prob = models.predict_proba(x_test)

    # Print the model name

    print('Model Name: ', models)

    # confusion matrix in sklearn

    from sklearn.metrics import multilabel_confusion_matrix
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import classification_report
    from math import sqrt

    print(confusion_matrix(y_pred, y_test)) # Verticle is actual values & horizontal is predicted values

    # Actual and predicted classes

    lst_actual_class = y_test
    lst_predicted_class = y_pred
    lst_predicted_prob_class = y_pred_prob

    # Class = Label 0-10

    lst_classes = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

    # Compute multi-class confusion matrix

    arr_out_matrix = multilabel_confusion_matrix(lst_actual_class, lst_predicted_class, labels=lst_classes)

    # Temp store results

    model_acc = [];
    model_recall = [];
    model_prec = [];
```

```
model_fscore = [];
model_spec = [];
model_bal_acc = [];
model_mcc = [];
for no_class in range(len(lst_classes)):
    arr_data = arr_out_matrix[no_class];
    print("Print Class: {0}".format(no_class));

    tp = arr_data[1][1]
    fn = arr_data[0][1]
    tn = arr_data[0][0]
    fp = arr_data[1][0]

    overall_tp+=tp
    overall_fn+=fn
    overall_fp+=fp
    overall_tn+=tn

    sensitivity = round(tp/(tp+fn), 3);
    specificity = round(tn/(tn+fp), 3);
    accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
    balanced_accuracy = round((sensitivity+specificity)/2, 3);

    precision = round(tp/(tp+fp), 3);
    f1Score = round((2*tp/(2*tp + fp + fn)), 3);

    mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
    MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
    model_acc.append(accuracy);
    model_prec.append(precision);
    model_recall.append(sensitivity);
    model_fscore.append(f1Score);
    model_spec.append(specificity);
    model_bal_acc.append(balanced_accuracy);
    model_mcc.append(MCC);
```

```
    print("TP={0}, FN={1}, TN={2}, FP={3}".format(tp, fn, tn, fp));
    print("Accuracy: {0}".format(accuracy));    # Accuracy score
    print("Precision: {0}".format(precision)); # Precision score
    print("Sensitivity: {0}".format(sensitivity)); # Recall score
    print("F1-Score: {0}".format(f1Score)); # F1 score
    print("Specificity: {0}".format(specificity)); # True Nagative Rate
    print("Balanced Accuracy: {0}".format(balanced_accuracy)); # Balance accuracy score
    print("MCC: {0}\n".format(MCC)); # Matthews Correlation Coefficient

# OVERALL - FINAL PREDICTION PERFORMANCE

# importing mean()

from statistics import mean
import math

accuracy=round(mean(model_acc)*100, 4)
precision=round(mean(model_prec)*100, 4)
sensitivity=round(mean(model_recall)*100, 4)
f1Score=round(mean(model_fscore), 4)
specificity=round(mean(model_spec)*100, 4)
balanced_accuracy=round(mean(model_bal_acc)*100, 4)
MCC=round(mean(model_mcc), 4)
print("Overall Performance Prediction:");
print("Accuracy: {0}%".format(round(mean(model_acc)*100, 4)));
print("Precision: {0}%".format(round(mean(model_prec)*100, 4)));
print("Recall or Sensitivity: {0}%".format(round(mean(model_recall)*100, 4)));
print("F1-Score: {0}".format(round(mean(model_fscore), 4)));
print("Specificity or True Nagative Rate: {0}%".format(round(mean(model_spec)*100, 4)));
print("Balanced Accuracy: {0}%\n".format(round(mean(model_bal_acc)*100, 4)));
print("MCC: {0}\n".format(round(mean(model_mcc), 4)))

overall_tp=overall_tp/10
overall_fn=overall_fn/10
```

```
overall_tn=overall_tn/10
overall_fp=overall_fp/10

overall_tp=round(overall_tp)
overall_tn=round(overall_tn)
overall_fp=round(overall_fp)
overall_fn=round(overall_fn)



fpr = {}
tpr = {}
thresh ={}

n_class = 10

for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(lst_actual_class, lst_predicted_prob_class[:,i], pos_label=i)

# plotting
plt.figure(figsize=(10,20))
plt.subplot(5,2,counter)
plt.plot(fpr[0], tpr[0], linestyle='--',color='brown', label='Class 0 vs Rest')
plt.plot(fpr[1], tpr[1], linestyle='--',color='green', label='Class 1 vs Rest')
plt.plot(fpr[2], tpr[2], linestyle='--',color='blue', label='Class 2 vs Rest')
plt.plot(fpr[3], tpr[3], linestyle='--',color='red', label='Class 3 vs Rest')
plt.plot(fpr[4], tpr[4], linestyle='--',color='purple', label='Class 4 vs Rest')
plt.plot(fpr[5], tpr[5], linestyle='--',color='yellow', label='Class 5 vs Rest')
plt.plot(fpr[6], tpr[6], linestyle='--',color='green', label='Class 6 vs Rest')
plt.plot(fpr[7], tpr[7], linestyle='--',color='blue', label='Class 7 vs Rest')
plt.plot(fpr[8], tpr[8], linestyle='--',color='magenta', label='Class 8 vs Rest')
plt.plot(fpr[9], tpr[9], linestyle='--',color='black', label='Class 9 vs Rest')

my_title="Multiclass ROC curve" + str(models)
plt.title(my_title)
plt.xlabel('False Positive Rate')
```

```
    plt.ylabel('True Positive rate')
    plt.legend(loc='best')

    # ROC AUC score - one-vs-one (OvO) algorithm computes the average of the ROC AUC scores for each class against
    # all other classes.

    print('roc_auc_score:', round(roc_auc_score(lst_actual_class, lst_predicted_prob_class, multi_class='ovo',
                                        average='weighted'),3))

    counter=counter+1

    #-----------------------------------------------------------------------------------------------------
    new_row = {'Model Name' : models,
            'True_Positive': overall_tp,
            'False_Negative': overall_fn,
            'False_Positive': overall_fp,
            'True_Negative': overall_tn,
            'Accuracy' : accuracy,
            'Precision' : precision,
            'Recall' : sensitivity,
            'F1 Score' : f1Score,
            'Specificity' : specificity,
            'MCC': MCC,
            'ROC_AUC_Score':round(roc_auc_score(y_test, y_pred_prob, multi_class='ovo', average='weighted'),3),
            'Balanced Accuracy':balanced_accuracy}
```

4.0 Conclusions and Future work

The model results in the following order by considering the model accuracy, F1 score and ROC AUC score.

1) **Light GBM**
2) **Gradient Boosting**
3) **Support Vector Machine Classification**

We recommend model – **Light GBM** with Stratified and Random Sampling technique as a best fit for the give n BI claims dataset. We considered Random Forest because it uses bootstrap aggregation which can reduce bias and variance in the data and can leads to good predictions with claims dataset.

	Model Name	True_Positive	False_Negative	False_Positive	True_Negative	Accuracy	Precision	Recall	F1 Score	Specificity
0	LogisticRegression()	778	722	722	12780	90.37	51.88	52.42	0.5148	94.67
1	DecisionTreeClassifier()	642	859	859	12643	88.54	42.71	43.05	0.4281	93.65
2	(DecisionTreeClassifier(max_features='sqrt', r...	801	700	700	12802	90.69	53.36	54.41	0.5334	94.82
3	(ExtraTreeClassifier(random_state=1675888055),...	786	714	714	12787	90.47	52.35	53.13	0.5236	94.71
4	KNeighborsClassifier()	710	790	791	12711	89.46	47.26	48.39	0.4753	94.15
5	SVC(probability=True)	830	671	671	12831	91.06	55.31	55.86	0.5489	95.06
6	(DecisionTreeClassifier(random_state=133744588...	791	709	709	12793	90.54	52.73	53.84	0.5279	94.75
7	([DecisionTreeRegressor(criterion='friedman_ms...	833	667	667	12835	91.12	55.49	56.43	0.5519	95.1
8	LGBMClassifier()	834	666	666	12836	91.14	55.6	56.61	0.5552	95.08
9	GaussianNB()	651	849	849	12653	88.69	43.55	47.51	0.4171	93.85

MCC	ROC_AUC_Score	Balanced Accuracy
0.4657	0.899	73.55
0.3649	0.677	68.36
0.4851	0.917	74.62
0.4733	0.913	73.92
0.4187	0.84	71.28
0.5038	0.916	75.46
0.4787	0.913	74.3
0.5076	0.92	75.76
0.5096	0.91	75.85
0.3804	0.845	70.68

5.0 References

The following links are referred in order to complete this project

- Music genre | Kaggle
- https://www.javatpoint.com/classification-algorithm-in-machine-learning
- https://www.ibm.com/docs/en/db2/9.7?topic=analytics-data-mining
- https://www.geeksforgeeks.org/machine-learning/#int

6.0 Appendices

6.1. Python code Results

Model Name: LogisticRegression()

Overall Performance Prediction:
Accuracy: 90.37%
Precision: 51.88%
Recall or Sensitivity: 52.42%
F1-Score: 0.5148
Specificity or True Negative Rate: 94.67%
Balanced Accuracy: 73.55%
MCC: 0.4657
roc_auc_score: 0.899

Model Name: DecisionTreeClassifier()

Overall Performance Prediction:
Accuracy: 88.54%
Precision: 42.71%
Recall or Sensitivity: 43.05%
F1-Score: 0.4281
Specificity or True Negative Rate: 93.65%
Balanced Accuracy: 68.36%
MCC: 0.3649
roc_auc_score: 0.677

Model Name: RandomForestClassifier()

Overall Performance Prediction:
Accuracy: 90.69%
Precision: 53.36%
Recall or Sensitivity: 54.41%
F1-Score: 0.5334
Specificity or True Negative Rate: 94.82%
Balanced Accuracy: 74.62%
MCC: 0.4851
roc_auc_score: 0.917

Model Name: ExtraTreesClassifier()

Overall Performance Prediction:
Accuracy: 90.47%
Precision: 52.35%
Recall or Sensitivity: 53.13%
F1-Score: 0.5236
Specificity or True Negative Rate: 94.71%
Balanced Accuracy: 73.92%
MCC: 0.4733
roc_auc_score: 0.913

Model Name: KNeighborsClassifier()
Overall Performance Prediction:

Accuracy: 89.46%
Precision: 47.26%
Recall or Sensitivity: 48.39%
F1-Score: 0.4753
Specificity or True Negative Rate: 94.15%
Balanced Accuracy: 71.28%
MCC: 0.4187
roc_auc_score: 0.84

Model Name: SVC(probability=True)

Overall Performance Prediction:
Accuracy: 91.06%
Precision: 55.31%
Recall or Sensitivity: 55.86%
F1-Score: 0.5489
Specificity or True Negative Rate: 95.06%
Balanced Accuracy: 75.46%
MCC: 0.5038
roc_auc_score: 0.916

Model Name: BaggingClassifier(n_estimators=100)

Overall Performance Prediction:
Accuracy: 90.54%
Precision: 52.73%
Recall or Sensitivity: 53.84%
F1-Score: 0.5279
Specificity or True Negative Rate: 94.75%
Balanced Accuracy: 74.3%
MCC: 0.4787
roc_auc_score: 0.913

Model Name: GradientBoostingClassifier()

Overall Performance Prediction:
Accuracy: 91.12%
Precision: 55.49%
Recall or Sensitivity: 56.43%
F1-Score: 0.5519
Specificity or True Nagative Rate: 95.1%
Balanced Accuracy: 75.76%
MCC: 0.5076
roc_auc_score: 0.92

Model Name: LGBMClassifier()

Overall Performance Prediction:
Accuracy: 91.14%
Precision: 55.6%
Recall or Sensitivity: 56.61%
F1-Score: 0.5552
Specificity or True Nagative Rate: 95.08%
Balanced Accuracy: 75.85%

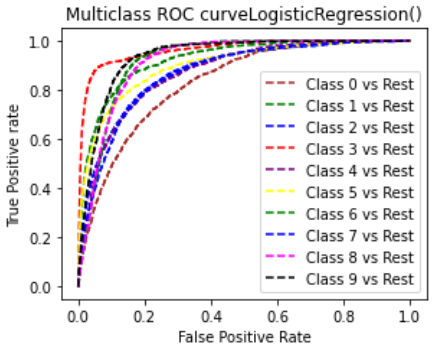
MCC: 0.5096
roc_auc_score: 0.91

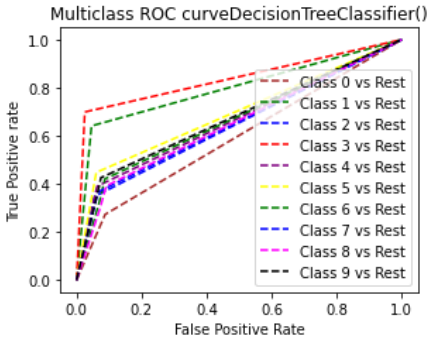
Model Name: GaussianNB()

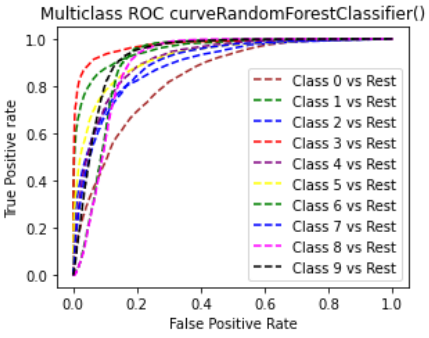
Overall Performance Prediction:
Accuracy: 88.69%
Precision: 43.55%
Recall or Sensitivity: 47.51%
F1-Score: 0.4171
Specificity or True Nagative Rate: 93.85%
Balanced Accuracy: 70.68%
MCC: 0.3804
roc_auc_score: 0.845

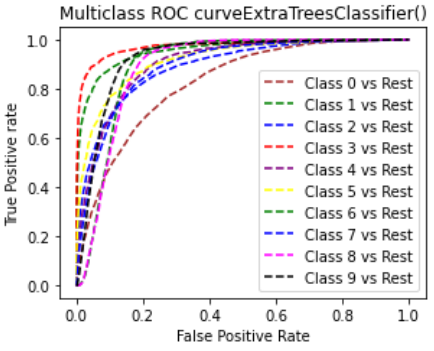
6.2. List of Charts

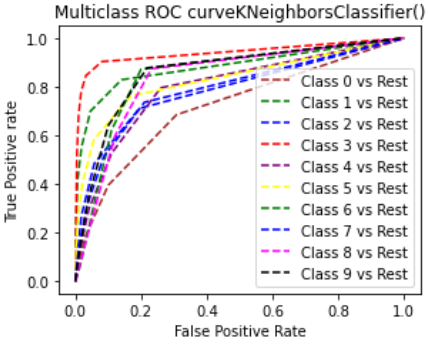
6.2.1 ROC curves for all Models

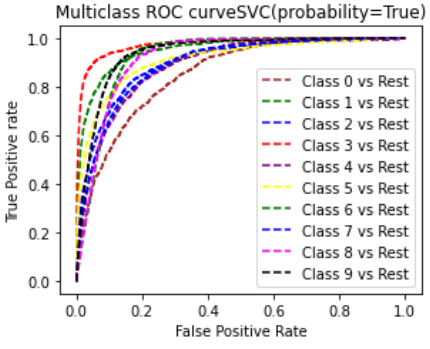


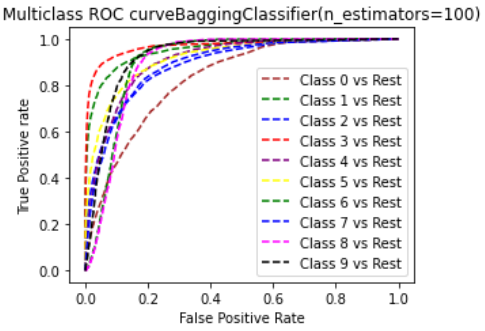


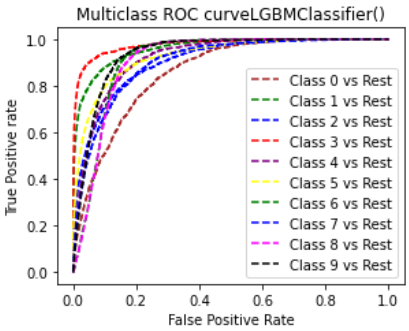


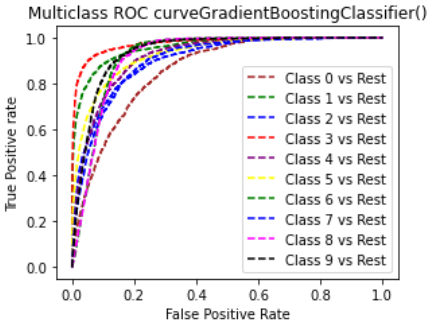


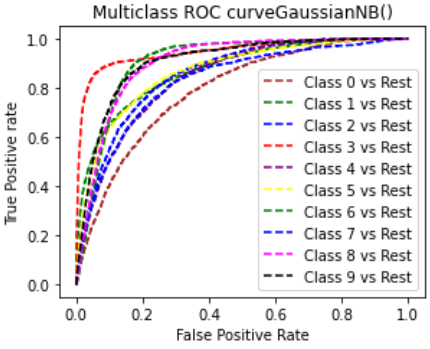





Multiclass ROC curveBaggingClassifier(n_estimators=100)
True Positive rate
False Positive Rate
Class 0 vs Rest
Class 1 vs Rest
Class 2 vs Rest
Class 3 vs Rest
Class 4 vs Rest
Class 5 vs Rest
Class 6 vs Rest
Class 7 vs Rest
Class 8 vs Rest
Class 9 vs Rest


Multiclass ROC curveLGBMClassifier()
True Positive rate
False Positive Rate
Class 0 vs Rest
Class 1 vs Rest
Class 2 vs Rest
Class 3 vs Rest
Class 4 vs Rest
Class 5 vs Rest
Class 6 vs Rest
Class 7 vs Rest
Class 8 vs Rest
Class 9 vs Rest


Multiclass ROC curveGradientBoostingClassifier()
True Positive rate
False Positive Rate
Class 0 vs Rest
Class 1 vs Rest
Class 2 vs Rest
Class 3 vs Rest
Class 4 vs Rest
Class 5 vs Rest
Class 6 vs Rest
Class 7 vs Rest
Class 8 vs Rest
Class 9 vs Rest


Multiclass ROC curveGaussianNB()
True Positive rate
False Positive Rate
Class 0 vs Rest
Class 1 vs Rest
Class 2 vs Rest
Class 3 vs Rest
Class 4 vs Rest
Class 5 vs Rest
Class 6 vs Rest
Class 7 vs Rest
Class 8 vs Rest
Class 9 vs Rest