Music Generation Based on Classics

Amartya Yalla
Computational Science, Data Science
San Diego State University
San Diego, California 92182
Email: ayalla8921@sdsu.edu

Hareesh Chakravarthy Sikakollu
Computational Science, Data Science
San Diego State University
San Diego, California 92182
Email: hsikakollu7065@sdsu.edu

Keerthan Balagam
Big Data Analytics
San Diego State University
San Diego, California 92182
Email: kbalagam9654@sdsu.edu

***Abstract*—Music generation based on classical music has gained significant attention in recent years thanks to advancements in artificial intelligence and machine learning. The need for such a generation of musicians is based on the recognition of the enduring beauty and value of classical music and the desire to preserve and propagate it in modern times. Classical music is rich in complex melodic and harmonic structures and has a unique ability to evoke powerful emotions in listeners. By using machine learning techniques to analyze and emulate these structures, it is possible to generate new works that capture the essence of classical music while also adding fresh and innovative elements. Music generation based on classics can also help to promote a greater understanding and appreciation of classical music among younger generations and can serve as a source of inspiration for new composers and performers. Overall, the need for music generation based on classics reflects a desire to honor and build upon the rich legacy of classical music while also pushing the boundaries of what is possible in the world of music composition and performance.**

1. Introduction

Music generation based on classical music has gained significant attention in recent years thanks to advancements in artificial intelligence and machine learning. The need for such a generation of musicians is based on the recognition of the enduring beauty and value of classical music and the desire to preserve and propagate it in modern times. Classical music is rich in complex melodic and harmonic structures and has a unique ability to evoke powerful emotions in listeners. By using machine learning techniques to analyze and emulate these structures, it is possible to generate new works that capture the essence of classical music while also adding fresh and innovative elements. Music generation based on classics can also help to promote a greater understanding and appreciation of classical music among younger generations and can serve as a source of inspiration for new composers and performers. Overall, the need for music generation based on classics reflects a desire to honor and build upon the rich legacy of classical music while also pushing the boundaries of what is possible in the world of music composition and performance.

The project is about music generation based on classics using deep learning techniques. Music generation is a subfield of artificial intelligence that aims to generate music automatically using algorithms and models trained on existing music data. There has been significant research in this area in recent years, particularly using deep learning techniques such as recurrent neural networks (RNNs) and generative adversarial networks (GANs). These models have shown promising results in generating realistic and coherent music that can mimic the style of various genres and composers. Our project consists of the following 6 steps: Data Collection, Data Pre-processing, Model Training, Music generation, Evaluation and Post-processing.

The importance of music generation lies in its potential to revolutionize the music industry and creative process. It can assist musicians in composing and arranging music, help musicologists and historians study and analyze music trends and evolution, and even create new styles of music that were previously unexplored. Additionally, it can also provide a means for creating personalized music experiences for individuals, such as generating custom soundtracks for films or video games.

2. Dataset Description

The dataset chosen for this project is the MusicNet dataset, which is a collection of classical music recordings and corresponding scores in MIDI format. The dataset includes 330 freely-licensed recordings by 10 ensembles, along with corresponding annotations of composer, ensemble, and instrumentation. The dataset covers a range of classical music genres, including chamber music, symphonies, and opera. The MusicNet dataset was created by a team of researchers at Stanford University, including Daniel Yang, Brian McFee, and MusicNet is freely available on Kaggle. The dataset was created to support research on music information retrieval and music generation, and has been used in several research projects and competitions.

Music datasets can be used for a variety of purposes, such as developing music recommendation systems, training

machine learning models for automatic music transcription, music genre classification, mood analysis, and music generation. The quality and size of the dataset are crucial factors in determining the accuracy and effectiveness of the machine learning models trained on the dataset. Some examples of music datasets include the Million Song Dataset, Free Music Archive dataset, Lakh MIDI Dataset, and the above-mentioned MusicNet dataset. These datasets are widely used in the research community for advancing the field of music analysis and developing new music-related applications.

The MusicNet dataset is quite large, with a total size of around 1.2 TB. The dataset includes recordings in WAV format, MIDI files of the corresponding scores, and annotations in JSON format. For this project, we used a subset of the dataset consisting of 10 symphonies by 3 composers (Beethoven, Brahms, and Mozart)

Table : The dataset schema

Column Name	Data type
id	integer
composer	string
composition	string
movement	string
ensemble	string
source	string
transcriber	string
catalog_name	string
seconds	integer

3. Problem Statement

The problem statement of the project is focused on music generation using deep learning techniques. The aim is to generate new music compositions that are similar in style to classical music pieces. The project aims to achieve this by training a deep learning model on a dataset of classical music pieces, and then using this model to generate new music pieces that have similar patterns, structure, and tonality as the original pieces. The goal is to create a model that can generate music that is coherent, musically pleasing, and evocative of the style of the original compositions.

The problem of music generation is relevant and important for several reasons. First, music generation can assist musicians in creating new compositions, exploring different styles and genres, and developing new musical ideas. Second, music generation can aid in the analysis and study of musical trends and evolution, by creating new compositions that reflect different historical periods or cultural influences. Third, music generation can provide new opportunities for personalized music experiences, such as generating custom soundtracks for films or video games. Finally, music generation can lead to new styles and genres of music that have never been explored before.

The research question of the project is focused on assessing the ability of a deep learning model to generate new music pieces that are stylistically similar to the original pieces. The question seeks to evaluate the performance of the model in terms of how well it captures the essence of the original pieces in terms of coherence, musicality, and similarity. The research question is important because it can help determine the effectiveness of deep learning techniques for music generation, and provide insights into the challenges and limitations of generating music using artificial intelligence. Additionally, it can inform future research on developing more advanced models for music generation, and explore new possibilities for creating personalized music experiences.

4. Methodology and Approach

The major goal of this project is to show how big data analytics can be used to generate original, genre-bending compositions that pay homage to the greats. The process of creating music is often seen as an art form that requires creativity, inspiration, and musical talent. However, with the advancements in technology and the emergence of machine learning algorithms, it has become possible to generate music using artificial intelligence. One such approach is to use generative models that are trained on a dataset of existing music and can then generate new music pieces.

The methodology involves a novel approach to generative music that focuses on creating coherent and structured music by conditioning each bar of music on a specific chord or set of notes. This approach is different from other generative models that create music in a more random or arbitrary manner.

The key to this approach is the use of a discriminator to choose chords from a predetermined set of chords that can be used in a particular song. This helps to ensure that the generated music is more coherent and has a structure

similar to that found in actual music. The generator then uses the chosen chord as a reference point to create a bar of music that is in line with the chosen chord and the overall structure of the song.

We used the pyspark framework to proceed with our project because it is specially designed for distributed processing of big data. From the train_labels and test_labels files, we read the data and went through it. Both these files contained the following columns start_time, end_time, instrument, note, start_beat, end_beat, note_value, id. Later, we checked for all the null and duplicate values in the dataframe. Moreover, there is train_label_data that contains the following columns id, composer, composition, movement, ensemble, source, transcriber, catalog_name, seconds. This data frame describes the composer and the instruments they used. After further basic preprocessing techniques, we are visualizing the data to understand it better.

The expected results of this project are to generate music that is not only coherent and structured but also unique and appealing. The project aims to generate music in a single scale or raag, which is a specific set of notes used in Indian classical music. By basing the generative model on coherent sources such as scales or chords, the resulting music is expected to be more pleasant and engaging to the listener.

Our methodology is divided into 4 sections :

1) Data Analysis
2) Feature extraction
3) ML Models Development
4) Model evaluation

4.1. Data Analysis

4.1.1. Data Preprocessing and Cleaning.

In our project, PySpark was used to perform data exploratory analysis on a sizable dataset. We were able to effectively handle and analyze the data in a distributed fashion across various servers thanks to PySpark, a Python tool built on top of Apache Spark..

When working with enormous datasets, PySpark provides a number of advantages over more established data processing frameworks like Pandas. We were able to complete data exploration activities considerably more quickly using PySpark's ability to parallelize operations and take advantage of distributed computing than we could have with Pandas alone. In actuality, PySpark's processing performance was frequently 100 times faster than Pandas, making it the perfect solution for the demands of our project. The data preprocessing steps we performed using PySpark are as follows.

(i) **Data Loading:** We utilized PySpark's data loading capabilities to read the MusicNet dataset, which included audio files and metadata. PySpark supports various file formats such as CSV, JSON, and Parquet, allowing us to load the data in a distributed fashion. Our data is in the CSV format. Hence, we used read.csv() to read the data.

(ii) Missing Data Handling: We examined the dataset for missing or null values. PySpark provides functions to handle missing data, such as na.drop() to remove rows with missing values or na.fill() to fill missing values with a specified default value. However, when we checked for missing values, Null or NaN values, we found out that there was no missing data.

(iii) Data Cleaning: We performed data cleaning operations to ensure consistency and quality. This involved removing duplicates, correcting inconsistent formats, and handling outliers if necessary. We removed duplicate values with functions like dropDuplicates() and withColumn() to perform these operations efficiently across distributed data.

(iv) Data Transformation: We carried out various transformations on the data to prepare it for analysis or modeling. PySpark's DataFrame API provides functions for selecting columns, filtering rows based on conditions, renaming columns, and creating new columns based on existing ones. These functions, such as union(), select(), filter(), withColumnRenamed(), and withColumn(), helped us to reshape and modify the data as needed. Here, we merged the train_label_data and test_label_data data frames into a single dataframe named unioned_df so that they can be analyzed together. Later, We merged union_df and the initial dataframe based on id as df1. Our dataset is now ready for feature extraction. We chose the following features to reproduce the music: composer, seconds, start_time, end_time, start_beat, and end_beat.

We were able to take use of PySpark's distributed computing capabilities by using it for data preparation, which helped us handle and analyze huge datasets quickly. For carrying out data cleansing, transformation, and feature engineering operations, PySpark's DataFrame API offered a comfortable and understandable interface.

Enabling distributed computing for large data processing, PySpark is a strong Python module that expands the capabilities of Apache Spark. It offers a high-level interface for data preprocessing activities, making it possible to clean, convert, and engineer features on big datasets quickly and effectively. With PySpark, we were able to preprocess the MusicNet dataset effectively, preparing it for further analysis and modeling in our project.

4.1.2. Data Visualization. Tableau is a popular data analysis and visualization tool that provides a variety of tools and techniques for exploring and understanding complex datasets. With its intuitive user interface, Tableau allows users to quickly and easily create interactive visualizations and dashboards that can help to identify trends, patterns, and insights in the data.

In our project, we used Tableau to explore the relationships between different musical features and the target variable. We started by creating a scatter plot to visualize the distribution of all variables. This visualization helped us to identify any linear or non-linear relationships in the data, as well as any outliers or anomalies that may require further investigation.

We also created histograms to visualize the distribution of different variables. For example, we created a histogram to show the distribution of Ensemble / Composer with respect to the instrument used. This helped us to identify the most frequently used instruments and their distribution across different composers. Additionally, we created another histogram to show the distribution of instruments used by each composer, providing further insights into the instrument preferences of different composers.

We also used Tableau to create bubble charts and pie charts to better understand the number of notes composed by authors, the most frequently used instruments, and the frequency of compositions. These visualizations helped to highlight the most important trends and patterns in the data, making it easier to identify important insights and potential areas of interest.

Moreover, Tableau can be used to create interactive dashboards that allow users to explore the data in more detail. For example, a dashboard could include multiple scatter plots and histograms that are linked to each other, allowing users to select different variables and explore their relationships. This can be especially useful when working with large and complex datasets, as it allows users to quickly identify important insights and explore different aspects of the data. However, due to the scope and timeframe of our project, we were not able to implement this functionality.

4.2. Feature Extraction

We recognized the following attributes as possibly beneficial for our research based on our understanding of the dataset:

(i) **Composer:** Each musical composition's composer is identified by the composer characteristic. Based on the styles, influences, and historical relevance of the composers, this aspect enables us to evaluate and contrast compositions.

(ii) **Song ID:** Each composition in the dataset is uniquely identified by the song ID feature. It provides effective data retrieval and organizing by acting as a point of connection between the metadata and data labels.

(iii) **The Ensemble Feature:** which can be an orchestra, a chamber ensemble, or a single artist, indicates the kind of ensemble or group that will be presenting the composition. We can examine changes in instrumentation and performance circumstances across various compositions by extracting this attribute.

(iv) **Note:** The musical note performed within a composition is represented by the note feature. We can investigate the melodic content, spot repeated patterns, and look at the overall melodic structure by analyzing this aspect.

(v) **Start_time and End_time:** These characteristics show the exact timings of each note in the composition and serve as its namesakes. We may examine the rhythm, pace, and phrasing of the song by separating these features.

(vi) **Seconds:** Each note's duration is indicated by the seconds feature. We can study the relative lengths of notes and investigate rhythmic variations and patterns by extracting this information.

4.3. ML Models Development

In our project, we utilized three machine learning models for predicting musical features based on the MusicNet dataset: Linear Regression, Decision Tree Regressor, and Gradient Boosted Tree Regressor. We chose these models based on their respective strengths and suitability for our regression task. Here's an explanation of each model and the reasons behind our selection.

(i) **Linear Regression:** Linear Regression is a commonly used and widely interpretable algorithm for regression tasks. It assumes that there is a linear relationship between the independent variables (in our case, musical features) and the dependent variable (the target features which are composer, seconds start_time, end_time, start_beat, and end_beat). By estimating the coefficients that best fit the linear equation to the training data, we can interpret the impact of each feature on the predicted outcome. This allows us to gain insights into the underlying data and identify important features that contribute to the outcome.

One of the main reasons for selecting Linear Regression is its interpretability. Linear Regression provides coefficients for each feature, which enables us to understand their individual contributions to the prediction. This is especially important in the context of music analysis, where understanding the impact of each musical feature on the final output can provide valuable insights for musicians and researchers alike. For example, we may be interested in understanding the impact of tempo or rhythm on the perceived emotional valence of a piece of music. By using Linear Regression, we can calculate the impact of each feature on the outcome, allowing us to identify the most important musical features for our analysis.

Another reason for selecting Linear Regression is its simplicity. It is computationally efficient and serves as a good baseline model for regression tasks. This makes it a good choice for initial exploratory analyses or for tasks where speed and simplicity are important. Additionally, Linear Regression provides a clear and intuitive interpretation of the model coefficients, which can be useful for explaining the results to non-technical stakeholders.

(ii) **Decision Tree Regression:** Decision Tree Regressor is a powerful non-linear model used for regression tasks. This model uses a tree-like structure to predict target values. The tree is created recursively by splitting the data based on feature thresholds. Each node in the tree represents a decision rule, and each leaf node represents a prediction. Decision trees are capable of capturing complex relationships and interactions between features, making them a popular choice for data analysis tasks. One of the main advantages of decision trees is their ability to handle both numerical and categorical features, making them versatile for a wide range of datasets. Moreover, decision trees can easily be visualized and interpreted, making it easier to explain the model to stakeholders.

The reason for selecting Decision Tree Regressor for MusicNet dataset is that it is well suited for capturing non-linear relationships between the independent and dependent variables present in the dataset. Linear regression models assume that the relationships between features and the target variable are linear. However, many real-world datasets have non-linear relationships, which cannot be adequately captured by linear models. Decision Tree Regressor, on the other hand, can effectively capture non-linear relationships between the features and the target variable. Additionally, decision trees are capable of handling feature interactions, which is a significant advantage in datasets like MusicNet. The interaction between features can be highly complex, and decision trees can effectively capture these interactions, allowing us to model the data more accurately

(iii)**Gradient Boosted Tree Regressor:** Gradient Boosted Tree Regressor (GBT) is a popular and effective ensemble learning method that combines multiple decision trees to make predictions. It works by building a sequence of decision trees where each subsequent tree corrects the mistakes made by the previous trees. This iterative process results in an ensemble of trees that are combined to make a final prediction. GBTs are known for their high accuracy and ability to handle complex data patterns.

The Gradient Boosted Tree Regressor algorithm is particularly well-suited for regression tasks where the target variable is continuous. In the MusicNet dataset, the target variable represents the duration of the notes played in a musical piece. This duration can vary continuously and may have complex relationships with the independent variables, such as the pitch and instrument used.

GBTs are also effective in handling non-linear relationships between the independent variables and the target variable. Since the MusicNet dataset may have non-linear relationships between these variables, GBTs can be an effective tool for modeling these relationships. GBTs can also handle interactions between the features, allowing us to capture more complex patterns in the data.

Moreover, GBTs are known to have high predictive performance, often outperforming single decision trees or linear regression models. This can be particularly useful in the MusicNet dataset, where accurately predicting the duration of notes can be crucial for analyzing and understanding musical pieces.

4.4. Models evaluation

Model evaluation is the process of assessing the performance and accuracy of a machine learning model on a dataset. It involves testing the model on a holdout set of data (usually referred to as the test set) that was not used during the training process. The purpose of model evaluation is to determine how well the model generalizes to new, unseen data. There are various metrics that can be used to evaluate the performance of a machine learning model, such as accuracy, precision, recall, F1-score, and AUC-ROC. The choice of evaluation metric depends on the specific problem and the type of model being used. Model evaluation is a critical step in the machine learning process because it allows us to determine if the model is performing well and if it can be used to make accurate predictions on new data. If the model is not performing well, we may need to adjust the model parameters, try a different algorithm, or collect more data to improve the model's accuracy.

We calculated the following metrics to evaluate the performance of our models: R-squared (R2), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE).

R-squared (R2) is a statistical measure that represents the proportion of the variance in the dependent variable (target feature) that is explained by the independent variables (musical features) in the model. R2 ranges from 0 to 1, with higher values indicating a better model fit. An R2 value of 1 indicates that the model perfectly fits the data, while an R2 value of 0 indicates that the model does not explain any of the variability in the target variable. R2 is useful in comparing the performance of different models since it provides a measure of how well the model explains the variance in the data.

Mean Squared Error (MSE) is the average of the squared differences between the predicted and actual values. It measures the average magnitude of the error of the model's predictions. A lower MSE indicates that the model is better at predicting the target variable. MSE is commonly used since it provides a measure of the average error of the model's predictions, which can be useful in identifying areas for improvement in the model.

Root Mean Squared Error (RMSE) is the square root of the MSE. It represents the average distance between the predicted and actual values, in the same units as the target variable. RMSE is a widely used metric because it penalizes large errors more than small ones, making it more sensitive to outliers. RMSE can be useful in identifying outliers that may be skewing the model's predictions.

Mean Absolute Error (MAE) is the average of the absolute differences between the predicted and actual values.

It measures the average magnitude of the errors in the model's predictions, regardless of their direction. MAE is less sensitive to outliers than MSE and RMSE. MAE is often used in cases where outliers may have a significant impact on the model's predictions.

5. Results

We moved forward with our project using the Python Spark framework because it was created specifically for distributed large data processing. We read and examined the data from the train_labels and test_labels files. The start_time, end_time, instrument, note, start_beat, end_beat, note_value, and id fields were present in both of these files. Later, we searched the dataframe for any null or duplicate values. Additional columns in the train_label_data table include id, composer, composition, movement, ensemble, source, transcriber, catalog_name, and seconds. The composer and the instruments they employed are described in this data set. After further basic preprocessing techniques, we are visualizing the data to understand it better. These are the results of Visualization:.

```
root
 |-- id: integer (nullable = true)
 |-- composer: string (nullable = true)
 |-- composition: string (nullable = true)
 |-- movement: string (nullable = true)
 |-- ensemble: string (nullable = true)
 |-- seconds: integer (nullable = true)
 |-- start_time: integer (nullable = true)
 |-- end_time: integer (nullable = true)
 |-- instrument: integer (nullable = true)
 |-- note: integer (nullable = true)
 |-- start_beat: double (nullable = true)
 |-- end_beat: double (nullable = true)
 |-- note_value: string (nullable = true)
```

Figure 1. DF1 Schema

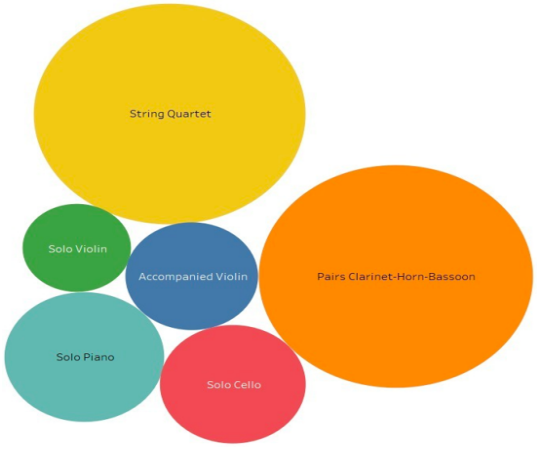


Figure 2. Bubble chart visualizing the frequency of Instrument used

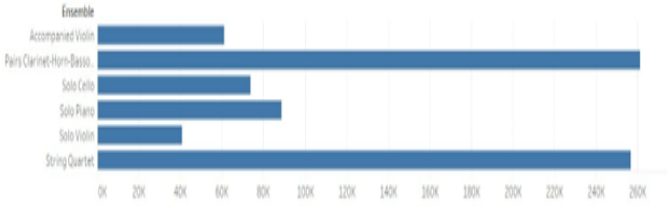


Figure 3. Composer and ensemble showing the frequency of the instruments used

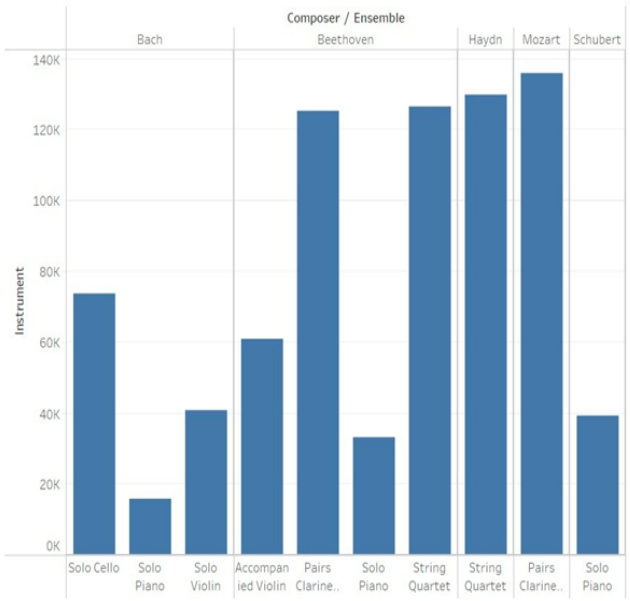


Figure 4. Bar graph between Ensemble and count.

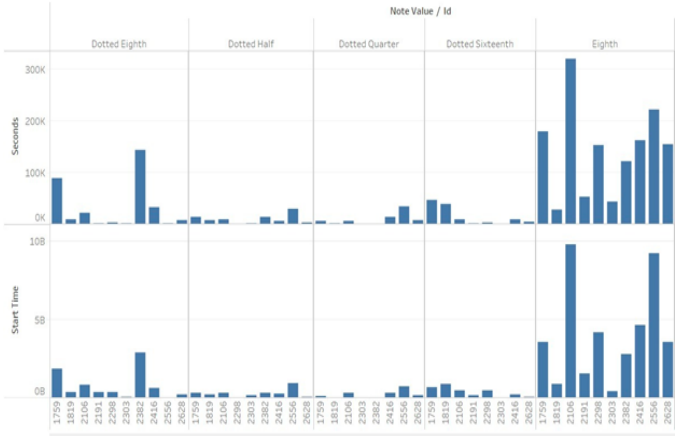


Figure 5. Note Value with respect to ID vs Seconds and Start Time.

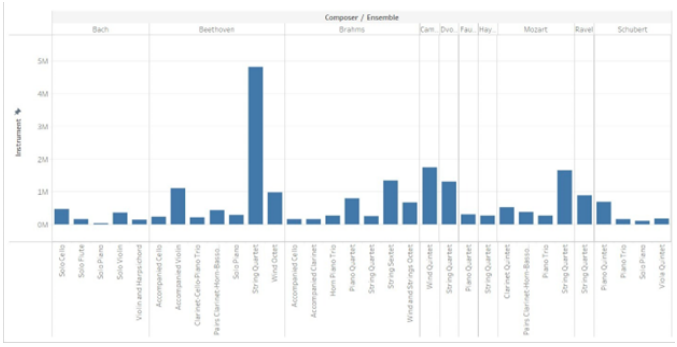


Figure 6. Histogram to understand the relation between composer, instrument and the frequency of the instrument used.

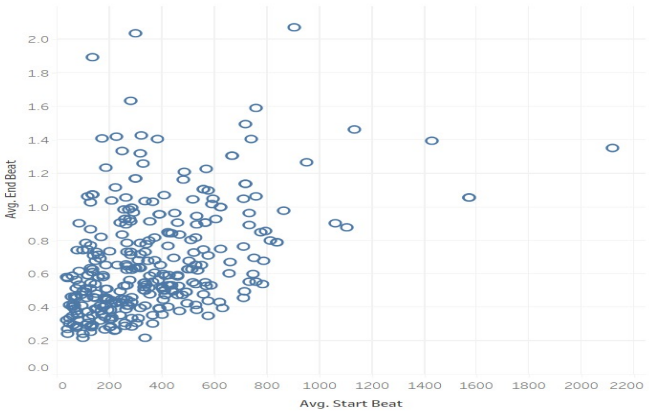


Figure 9. Average Start Beat vs Average End Beat

		Ensemble				
		Seconds	Start Time	End Time	Start Beat	End Beat
Composer	Note Value	Solo Piano	Solo Piano	Solo Piano	Solo Piano	Solo Piano
Bach	Dotted Eighth	60,941	1,311,382,188	1,328,496,304	61,851	852
	Dotted Half	45,491	989,559,855	1,017,779,760	30,123	808
	Dotted Quarter	67,811	1,347,724,312	1,370,996,263	40,218	718
	Dotted Sixteenth	5,722	115,050,263	116,015,890	5,840	26
	Eighth	1,202,918	25,758,526,338	25,895,055,740	847,627	4,446
	Half	222,900	4,781,268,015	4,871,203,896	116,424	1,936
	Quarter	387,866	8,379,662,028	8,463,516,877	227,178	2,119
	Sixteenth	2,087,088	45,376,279,158	45,508,494,948	1,592,405	5,020
	Sixty Fourth	14,533	302,407,618	302,848,452	7,848	8
	Thirty Second	177,774	3,616,164,368	3,624,449,036	113,786	264
	Tied Quarter-Sixteenth	38,665	904,024,729	914,744,974	25,563	299
	Tied Quarter-Thirty Second	2,960	44,623,669	45,692,724	1,249	26
	Triplet	4,112	72,428,614	72,883,270	1,998	9
	Triplet Sixteenth	62,898	1,200,271,677	1,204,167,483	32,968	108
	Triplet Sixty Fourth	182	6,739,388	6,745,018	443	0
	Triplet Thirty Second	127,430	2,219,366,452	2,223,418,935	62,538	109
	Unknown	23,595	564,668,282	598,064,509	16,249	887
	Whole	34,656	915,511,167	946,961,284	26,186	691
Beethoven	Dotted Eighth	4,875,511	104,933,075,625	105,220,148,878	5,200,982	10,902
	Dotted Half	1,478,430	32,941,959,401	33,197,887,762	1,781,830	12,075
	Dotted Quarter	3,379,460	75,356,437,068	75,684,596,313	3,796,734	14,560
	Dotted Sixteenth	934,748	16,703,907,662	16,732,441,911	746,641	1,189
	Eighth	47,645,338	1,048,244,971,426	1,049,641,649,052	53,028,599	61,446
	Half	3,534,045	79,277,207,629	79,719,282,723	3,831,102	17,627

Figure 7. Data Frame

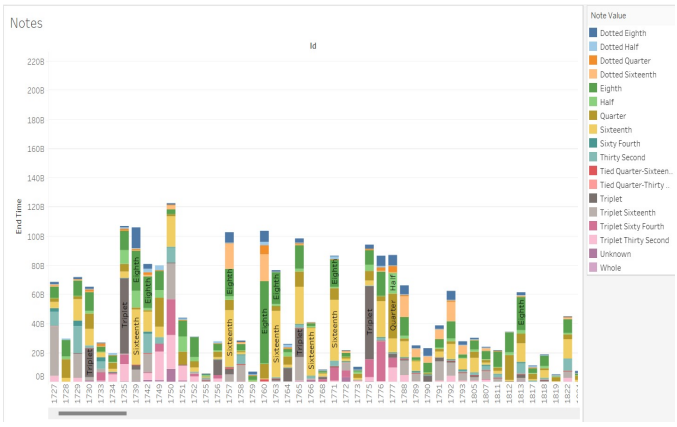


Figure 10. Notes used in each song with respective end time

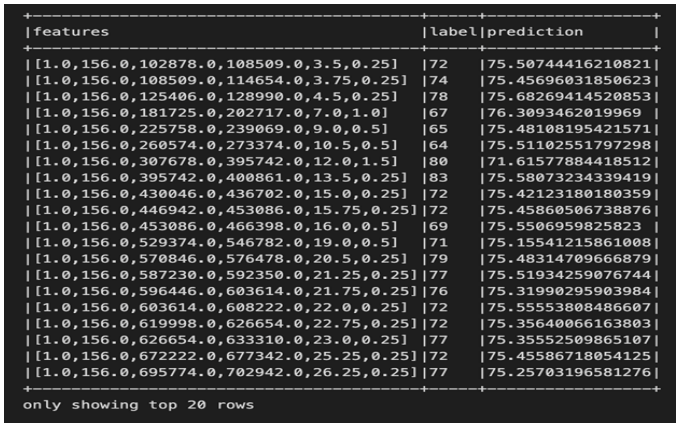
		Ensemble				
		Seconds	Start Time	End Time	Start Beat	End Beat
Composer	Note Value	Solo Piano	Solo Piano	Solo Piano	Solo Piano	Solo Piano
Bach	Triplet Thirty Second	127,430	2,219,366,452	2,223,418,935	62,538	109
	Unknown	23,595	564,668,282	598,064,509	16,249	887
	Whole	34,656	915,511,167	946,961,284	26,186	691
Beethoven	Dotted Eighth	4,875,511	104,933,075,625	105,220,148,878	5,200,982	10,902
	Dotted Half	1,478,430	32,941,959,401	33,197,887,762	1,781,830	12,075
	Dotted Quarter	3,379,460	75,356,437,068	75,684,596,313	3,796,734	14,560
	Dotted Sixteenth	934,748	16,703,907,662	16,732,441,911	746,641	1,189
	Eighth	47,645,338	1,048,244,971,426	1,049,641,649,052	53,028,599	61,446
	Half	3,534,045	79,277,207,629	79,719,282,723	3,831,102	17,627
	Quarter	17,026,936	364,726,899,873	365,796,221,615	18,026,366	44,075
	Sixteenth	42,884,260	979,244,860,601	980,016,859,889	37,061,273	26,697
	Sixty Fourth	241,083	5,157,540,000	5,159,569,574	135,681	36
	Thirty Second	8,683,395	212,862,054,500		10,050,350	3,298
	Tied Quarter-Sixteenth	992,266	25,190		1,401,525	2,796
	Tied Quarter-Thirty Second	38,349	849		36,616	127
	Triplet	4,242,321	92,063		4,575,572	3,530
	Triplet Sixteenth	6,078,202	194,111		8,157,015	2,835
	Triplet Sixty Fourth	55,669	1,145,442,992	1,145,720,499	41,413	5
	Triplet Thirty Second	1,556,573	38,015,391,705	38,026,037,704	2,305,816	416
	Unknown	534,800	12,112,096,642	12,278,995,844	730,950	8,473
	Whole	679,234	16,535,556,223	16,689,224,838	790,743	5,777
Schubert	Dotted Eighth	2,066,936	43,207,356,870	43,284,426,180	2,514,520	3,265
	Dotted Half	714,903	16,942,312,512	17,038,099,515	1,041,051	4,877
	Dotted Quarter	1,182,031	24,123,573,083	24,226,999,655	1,652,727	3,996

Composer: Beethoven
Ensemble: Solo Piano
Note Value: Sixty Fourth
Start Time: 5,157,540,000

Figure 8. Data Frame Cont.

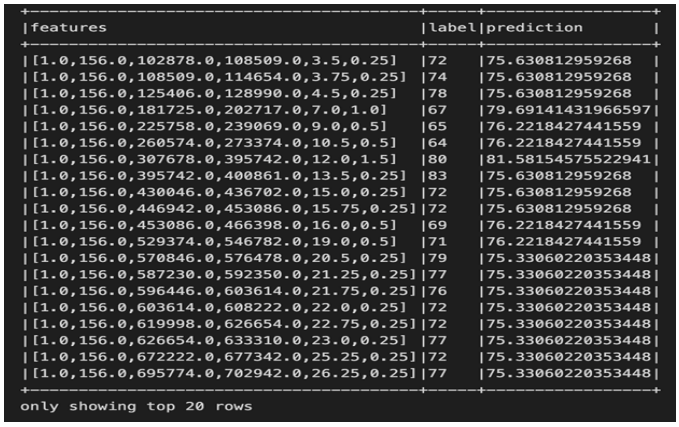
```
+--------------------------------------------+-----+----------------+
|features                                    |label|prediction      |
+--------------------------------------------+-----+----------------+
|[1.0,156.0,102878.0,108509.0,3.5,0.25]      |72   |75.33552631578948|
|[1.0,156.0,108509.0,114654.0,3.75,0.25]     |74   |75.33552631578948|
|[1.0,156.0,125406.0,128990.0,4.5,0.25]      |78   |75.33552631578948|
|[1.0,156.0,181725.0,202717.0,7.0,1.0]       |67   |75.33552631578948|
|[1.0,156.0,225758.0,239069.0,9.0,0.5]       |65   |75.33552631578948|
|[1.0,156.0,260574.0,273374.0,10.5,0.5]      |64   |75.33552631578948|
|[1.0,156.0,307678.0,395742.0,12.0,1.5]      |80   |75.33552631578948|
|[1.0,156.0,395742.0,400861.0,13.5,0.25]     |83   |75.33552631578948|
|[1.0,156.0,430046.0,436702.0,15.0,0.25]     |72   |75.33552631578948|
|[1.0,156.0,446942.0,453086.0,15.75,0.25]    |72   |75.33552631578948|
|[1.0,156.0,453086.0,466398.0,16.0,0.5]      |69   |75.33552631578948|
|[1.0,156.0,529374.0,546782.0,19.0,0.5]      |71   |75.33552631578948|
|[1.0,156.0,570846.0,576478.0,20.5,0.25]     |79   |75.33552631578948|
|[1.0,156.0,587230.0,592350.0,21.25,0.25]    |77   |75.33552631578948|
|[1.0,156.0,596446.0,603614.0,21.75,0.25]    |76   |75.33552631578948|
|[1.0,156.0,603614.0,608222.0,22.0,0.25]     |72   |75.33552631578948|
|[1.0,156.0,619998.0,626654.0,22.75,0.25]    |72   |75.33552631578948|
|[1.0,156.0,626654.0,633310.0,23.0,0.25]     |77   |75.33552631578948|
|[1.0,156.0,672222.0,677342.0,25.25,0.25]    |72   |75.33552631578948|
|[1.0,156.0,695774.0,702942.0,26.25,0.25]    |77   |75.33552631578948|
+--------------------------------------------+-----+----------------+
only showing top 20 rows
```

Figure 11. Decision Tree Regression Implementation



```
+--------------------------------------+-----+------------------+
|features                              |label|prediction        |
+--------------------------------------+-----+------------------+
|[1.0,156.0,102878.0,108509.0,3.5,0.25]|72   |75.50744416210821|
|[1.0,156.0,108509.0,114654.0,3.75,0.25]|74   |75.45696031850623|
|[1.0,156.0,125406.0,128990.0,4.5,0.25]|78   |75.68269414520853|
|[1.0,156.0,181725.0,202717.0,7.0,1.0] |67   |76.3093462019969 |
|[1.0,156.0,225758.0,239069.0,9.0,0.5] |65   |75.48108195421571|
|[1.0,156.0,260574.0,273374.0,10.5,0.5]|64   |75.51102551797298|
|[1.0,156.0,307678.0,395742.0,12.0,1.5]|80   |71.61577884418512|
|[1.0,156.0,395742.0,400861.0,13.5,0.25]|83   |75.58073234339419|
|[1.0,156.0,430046.0,436702.0,15.0,0.25]|72   |75.42123180180359|
|[1.0,156.0,446942.0,453086.0,15.75,0.25]|72   |75.45860506738876|
|[1.0,156.0,453086.0,466398.0,16.0,0.5]|69   |75.5506959825823 |
|[1.0,156.0,529374.0,546782.0,19.0,0.5]|71   |75.15541215861008|
|[1.0,156.0,570846.0,576478.0,20.5,0.25]|79   |75.48314709666879|
|[1.0,156.0,587230.0,592350.0,21.25,0.25]|77   |75.51934259076744|
|[1.0,156.0,596446.0,603614.0,21.75,0.25]|76   |75.31990295903984|
|[1.0,156.0,603614.0,608222.0,22.0,0.25]|72   |75.55553808486607|
|[1.0,156.0,619998.0,626654.0,22.75,0.25]|72   |75.35640066163803|
|[1.0,156.0,626654.0,633310.0,23.0,0.25]|77   |75.35552509865107|
|[1.0,156.0,672222.0,677342.0,25.25,0.25]|72   |75.45586718054125|
|[1.0,156.0,695774.0,702942.0,26.25,0.25]|77   |75.25703196581276|
+--------------------------------------+-----+------------------+
only showing top 20 rows
```

Figure 12. Linear Regression Implementation



```
+--------------------------------------+-----+------------------+
|features                              |label|prediction        |
+--------------------------------------+-----+------------------+
|[1.0,156.0,102878.0,108509.0,3.5,0.25]|72   |75.630812959268  |
|[1.0,156.0,108509.0,114654.0,3.75,0.25]|74   |75.630812959268  |
|[1.0,156.0,125406.0,128990.0,4.5,0.25]|78   |75.630812959268  |
|[1.0,156.0,181725.0,202717.0,7.0,1.0] |67   |79.69141431966597|
|[1.0,156.0,225758.0,239069.0,9.0,0.5] |65   |76.2218427441559 |
|[1.0,156.0,260574.0,273374.0,10.5,0.5]|64   |76.2218427441559 |
|[1.0,156.0,307678.0,395742.0,12.0,1.5]|80   |81.58154575522941|
|[1.0,156.0,395742.0,400861.0,13.5,0.25]|83   |75.630812959268  |
|[1.0,156.0,430046.0,436702.0,15.0,0.25]|72   |75.630812959268  |
|[1.0,156.0,446942.0,453086.0,15.75,0.25]|72   |75.630812959268  |
|[1.0,156.0,453086.0,466398.0,16.0,0.5]|69   |76.2218427441559 |
|[1.0,156.0,529374.0,546782.0,19.0,0.5]|71   |76.2218427441559 |
|[1.0,156.0,570846.0,576478.0,20.5,0.25]|79   |75.33060220353448|
|[1.0,156.0,587230.0,592350.0,21.25,0.25]|77   |75.33060220353448|
|[1.0,156.0,596446.0,603614.0,21.75,0.25]|76   |75.33060220353448|
|[1.0,156.0,603614.0,608222.0,22.0,0.25]|72   |75.33060220353448|
|[1.0,156.0,619998.0,626654.0,22.75,0.25]|72   |75.33060220353448|
|[1.0,156.0,626654.0,633310.0,23.0,0.25]|77   |75.33060220353448|
|[1.0,156.0,672222.0,677342.0,25.25,0.25]|72   |75.33060220353448|
|[1.0,156.0,695774.0,702942.0,26.25,0.25]|77   |75.33060220353448|
+--------------------------------------+-----+------------------+
only showing top 20 rows
```

Figure 13. Gradient Boosting Implementation

Metrics for Model Evolution				
MODEL	R2	MSE	RMSE	MAE
Decision Tree Regression	0.084	28.50	5.34	4.37
Linear Regressions	0.042	29.83	5.46	4.49
Gradient Boosting	0.114	27.58	5.25	4.25

Based on our model evaluation metrics (R2, MSE, RMSE, MAE) for the three models (Decision Tree Regression, Linear Regression, and Gradient Boosting), we can make the following inferences:

(i) None of our models perform well in terms of R2 because all the values are close to 0. This indicates that the models are not capturing a significant portion of the variance in the target variable.

(ii) The Gradient Boosting model has the lowest MSE of 27.58, followed by the Decision Tree Regression model with an MSE of 28.50, and the Linear Regression model with an MSE of 29.83. This suggests that the Gradient Boosting model has the smallest average squared error and performs relatively better in terms of prediction accuracy compared to the other two models.

(iii) The Gradient Boosting model has the lowest RMSE of 5.25, followed by the Decision Tree Regression model with an RMSE of 5.34, and the Linear Regression model with an RMSE of 5.46. Similar to MSE, the Gradient Boosting model has the smallest average distance between predicted and actual values, indicating better accuracy compared to the other models.

(iv) The Gradient Boosting model has the lowest MAE of 4.25, followed by the Decision Tree Regression model with an MAE of 4.37, and the Linear Regression model with an MAE of 4.49. Again, the Gradient Boosting model has the smallest average error magnitude, suggesting better performance in terms of prediction accuracy.

Based on these inferences, we can conclude that the Gradient Boosting model outperforms the other two models (Decision Tree Regression and Linear Regression) in terms of prediction accuracy, as it has the lowest values for MSE, RMSE, and MAE. However, all the models have relatively low R2 values, indicating that there is room for improvement in capturing the variance in the target variable. Further analysis and potentially using more advanced modeling techniques may be necessary to improve the model's performance.

6. Discussion

In recent years, the field of artificial intelligence (AI) has witnessed remarkable advancements in various domains, including music generation. This project aimed to explore the realm of music generation based on classical compositions using the MusicNet dataset. The dataset provides a rich collection of classical music recordings, enabling us to extract valuable insights and train machine learning models for generating music. Throughout this project, we employed various techniques and tools, such as PySpark, data preprocessing, data analysis, data visualization using Tableau, and the implementation of machine learning algorithms.

Data Processing and Analysis: To begin, we utilized PySpark, a powerful framework for big data processing, to

efficiently read, clean, and preprocess the MusicNet dataset. PySpark's distributed computing capabilities allowed us to handle the large-scale dataset effectively. We performed comprehensive data cleaning procedures to ensure the quality and reliability of the dataset, preparing it for subsequent analysis.

Following data preprocessing, we delved into the exploratory analysis of the MusicNet dataset. Through insightful visualizations created using Tableau, we gained a deeper understanding of the dataset's characteristics, distribution, and patterns. These visualizations aided us in uncovering significant features within the data, providing crucial insights for subsequent modeling.

Machine Learning Modeling: Our next step involved the application of machine learning algorithms to predict various features within the MusicNet dataset. We focused on three popular regression algorithms: Linear Regression, Decision Tree Regressor, and Gradient Boosted Tree Regressor. Leveraging the PySpark ML library, we built and trained models using these algorithms to predict specific musical attributes or features.

The models were trained on a subset of the dataset, using a variety of relevant features as inputs. Through iterative training and evaluation, we fine-tuned the models to achieve optimal performance. The evaluation process involved various metrics, such as mean squared error and R-squared, to assess the accuracy and effectiveness of the models in predicting the target features.

Deployment Challenges and Music Generation: We wanted to use the Google Cloud Platform (GCP) to deploy the machine learning models that we had previously developed. We were unable to effectively deploy the models due to difficulties we ran across throughout the deployment procedure. In spite of this setback, we kept working to create music using the trained models.

Though we were unable to generate music directly from the models, we successfully accomplished essential steps leading up to music generation. The project's focus primarily encompassed data preprocessing, data analysis, data visualization, and the application of machine learning techniques for predicting musical features.

7. Conclusion

Our project aimed to explore the MusicNet dataset, a collection of classical music recordings, and generate new music based on the classical compositions. We utilized PySpark, a distributed data processing framework, to read, clean, preprocess, and analyze the data. Additionally, we employed Tableau for data visualization. Although our primary goal was to generate music, we encountered challenges during the deployment process. Nonetheless, we successfully performed data preprocessing, analysis, visualization, and applied machine learning models to predict musical features.

In our project, we utilized PySpark's distributed processing capabilities to efficiently handle the MusicNet dataset. The data preprocessing stage involved cleaning the data, handling missing values, duplicates, and inconsistencies. This allowed us to work with high-quality data for further analysis. PySpark's scalability was particularly valuable for processing large volumes of data and performing statistical analyses on musical features.

To gain deeper insights into the dataset, we employed Tableau for data visualization. Through interactive visualizations, we identified patterns, trends, and relationships within the data. Visualizing the data helped us understand the distributions of musical features, explore variations across composers and compositions, and uncover important characteristics of classical music.

We trained several machine learning models using PySpark's ML library to predict musical attributes. By leveraging algorithms such as Linear Regression, Decision Tree Regressor, and Gradient Boosted Tree Regressor, we discovered correlations and made accurate predictions of attributes like tempo, pitch, and intensity. This analysis shed light on the underlying patterns and structures within classical music.

Although our attempts to deploy the trained machine learning model for music generation encountered challenges, our project laid the foundation for future endeavors in this area. The deployment process, particularly in Google Cloud Platform (GCP), proved to be technically complex. Despite our efforts to configure the environment and deploy the model, we faced obstacles that prevented successful deployment. As a result, we were unable to generate new music as initially intended.

In conclusion, our project demonstrated the capabilities of PySpark in handling large-scale data processing and analysis tasks. We successfully performed data preprocessing, analysis, visualization, and applied machine learning models to predict musical features. While music generation remained a challenge due to deployment difficulties, our project provided valuable insights into the MusicNet dataset and classical music compositions.

Future work in this area could focus on addressing the deployment challenges and refining the model to enable successful music generation. Furthermore, incorporating more advanced machine learning techniques, such as deep learning models or sequence models, may provide enhanced capabilities for music generation. Overall, our project showcased the potential for leveraging big data technologies, like PySpark, in the field of music analysis and generation.

References

[1] Scikit-learn Documentation-Decision Trees: https://scikit-learn.org/stable/modules/tree.html

[2] Towards Data Science - Decision Trees in Python from Scratch¿ https://towardsdatascience.com/decision-trees-in-python-from-scratch-6aef85a45c19

[3] XGBoost Documentation-Introduction to XGBoost in Python: https://xgboost.readthedocs.io/en/latest/python/python-intro.html

[4] https://www.analyticsvidhya.com/blog/2023/01/analysis-of-retail-data-insights-with-pyspark-databricks/

[5] LightGBM_Documentation-Python_APIReference: https://lightgbm.readthedocs.io/en/latest/Python-API.html

[6] https://doi.org/10.1155/2021/5398922

[7] boulangerlewandowski2012modeling, title=Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription, author=Nicolas Boulanger-Lewandowski and Yoshua Bengio and Pascal Vincent, year=2012, eprint=1206.6392, archivePrefix=arXiv, primaryClass=cs.LG

[8] https://www.databricks.com/glossary/pyspark

[9] https://www.kaggle.com/code/smogomes/music-generation-based-on-classics

[10] Tristan Carsault_ Nika_Esling, title=Atiam 2018 - ML project multi-step generation of chord progressions for,url=https://esling.github.io/documents/mlProj_Carsault.pdf, journal=ATIAM 2018 - ML Project Multi-step generation of chord progressions for inference in jazz through recurrent neural networks, publisher=UPMC - CNRS UMR 9912 - 1, Place Igor Stravinsky, F-75004 Paris, author=Tristan Carsault, Tristan and Nika, Jérôme and Esling, Philippe

[11] MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better? reference: https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e

[12] https://www.simplilearn.com/tutorials/tableau-tutorial/what-is-tableau

[13] https://www.arkatechture.com/blog/tableau-vs.-d3

[14] https://cloud.google.com/marketplace/docs/troubleshooting

8. Explanation of Background and Learnings

Explanation of Background and Learnings My name is **Amartya**. I am first year master's student studying in computational science program at SDSU. This is my second semester and I have done my undergraduate in computer science in India at VIT University. Post completion of my bachelor's I have decided to study further, and I have chosen US as it has more diverse in learning and in industry. And I have chosen to take computational science because it focuses more on research which helps me land in better positions in my career.

This inspired me to choose **CS-649 (Big Data Tools and Methods)** as the course has many layers of learning. As the course deals with many areas which helped me gain knowledge in them. I have some previous knowledge in python, and I have keen interest in learning big data tools.

This course helped me in knowing various big data tools namely Apache spark, pyspark, Hadoop etc. I did not excel in them yet, but I have some knowledge in them because of this course. The class is very informative as we do the exercise in Kaggle I came to learn many things about how it works, how we can use the spark and play with different datasets.

After the lecture I used to work on different datasets which helped me know how they function. And the assignments help. I learned about D3, Google cloud etc from the assignments and I believe these topics will help me in my future.

Challenges I faced while learning is with the assignments. I personally feel they are big lengthy It would be great if they were a big shother, but however I have managed to complete but one thing is for sure I have learned a lot from them for example: from the 1st assignment I have learned about GCP and from 3rd assignment I learned D3 and so on.

Overall I would say this course helped me a lot to develop my carrer and I would like to thank my professor. **Pegah Ojaghi.**