Introduction:

Problem Identification / Statement: The project aims to quantitatively analyze and predict team and player performances in the Valorant Champions Tour 2023, with the goal of improving win rates. Success will be measured by achieving a predictive accuracy of at least 75% in match outcome predictions and identifying possible team compositions to ensure success of both professional and casual players.

Context: Esports, particularly competitive games like Valorant, are rich in data and ripe for analytical exploration. Teams and players are constantly looking for insights to gain a competitive edge. The esports industry, including sponsors and team managers, can leverage these insights for decision-making and strategy development.

Data Source: The data contains around 6000 rows with around 20 columns. The provided dataset with match IDs, game IDs, team names, scores, player IDs, player names, agents, and various performance metrics like ACS, kills, deaths, assists, etc.

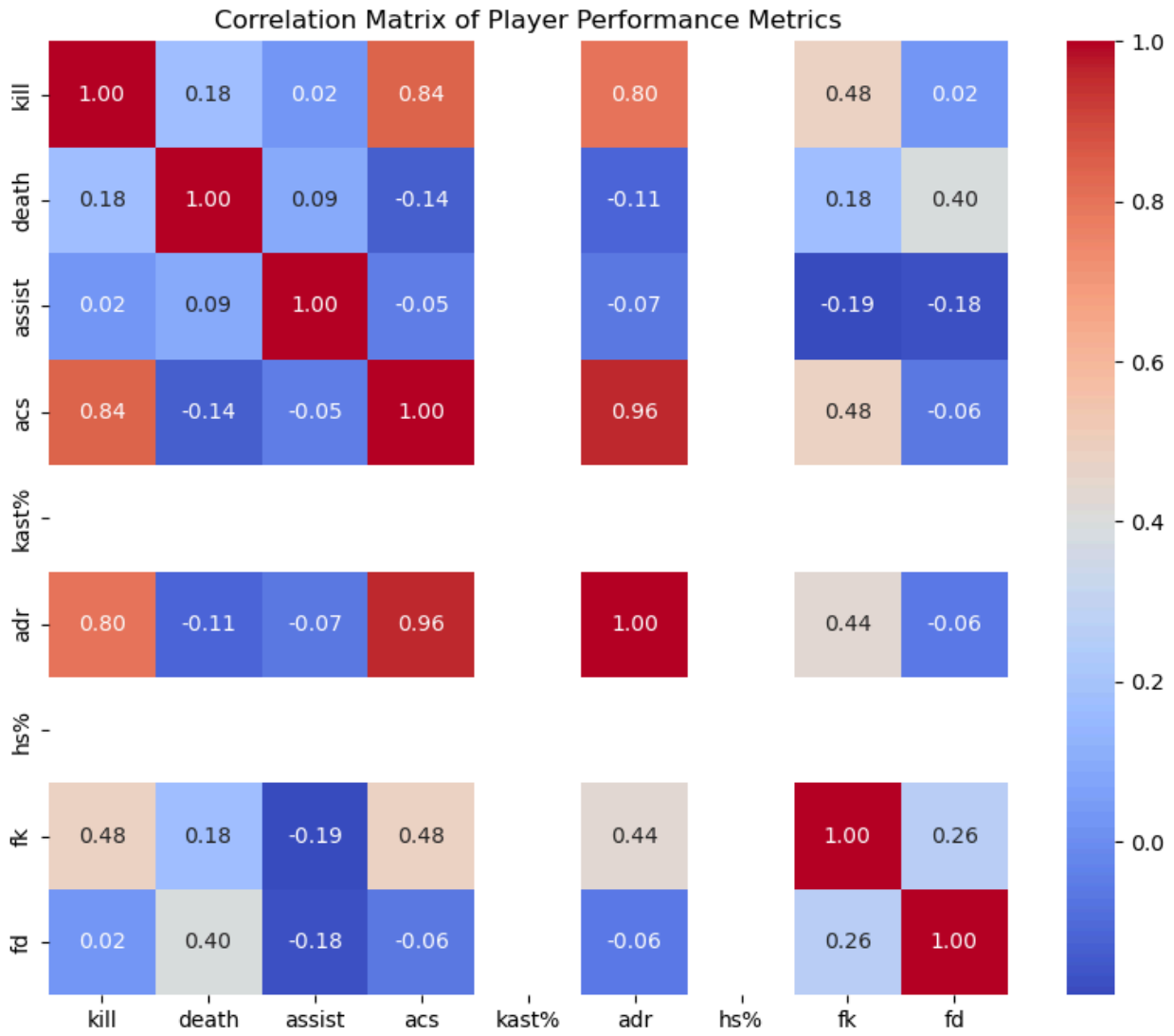
Background: Valorant is a team-based shooter game where players assume the role of agents, each with unique abilities. Matches are played on various maps, requiring strategy and skill. Agents contribute to their team's success through kills, assists, and strategic play. Performance is often measured by metrics like KDA (Kills, Deaths, Assists) and ACS (Average Combat Score). Competitive matches typically follow a best-of-three format, with the winning team needing to secure victories in two out of three games to win the match. This setup tests teams' adaptability and strategy across different game environments.

Methodology (Data Science Pipeline):

Note: Instead of splitting the classes into steps of the data science pipeline, data wrangling was done in the main class, feature engineering was done with the models, EDA was its own separate class, and preprocessing instead of having a separate class was done in the traditional (WinPredictionModel), deep learning, and recommendation system classes. This was to keep the readability of the code simpler, and to show the thought process of the code. Also problem identification was done above, so this section will start with Data Wrangling.

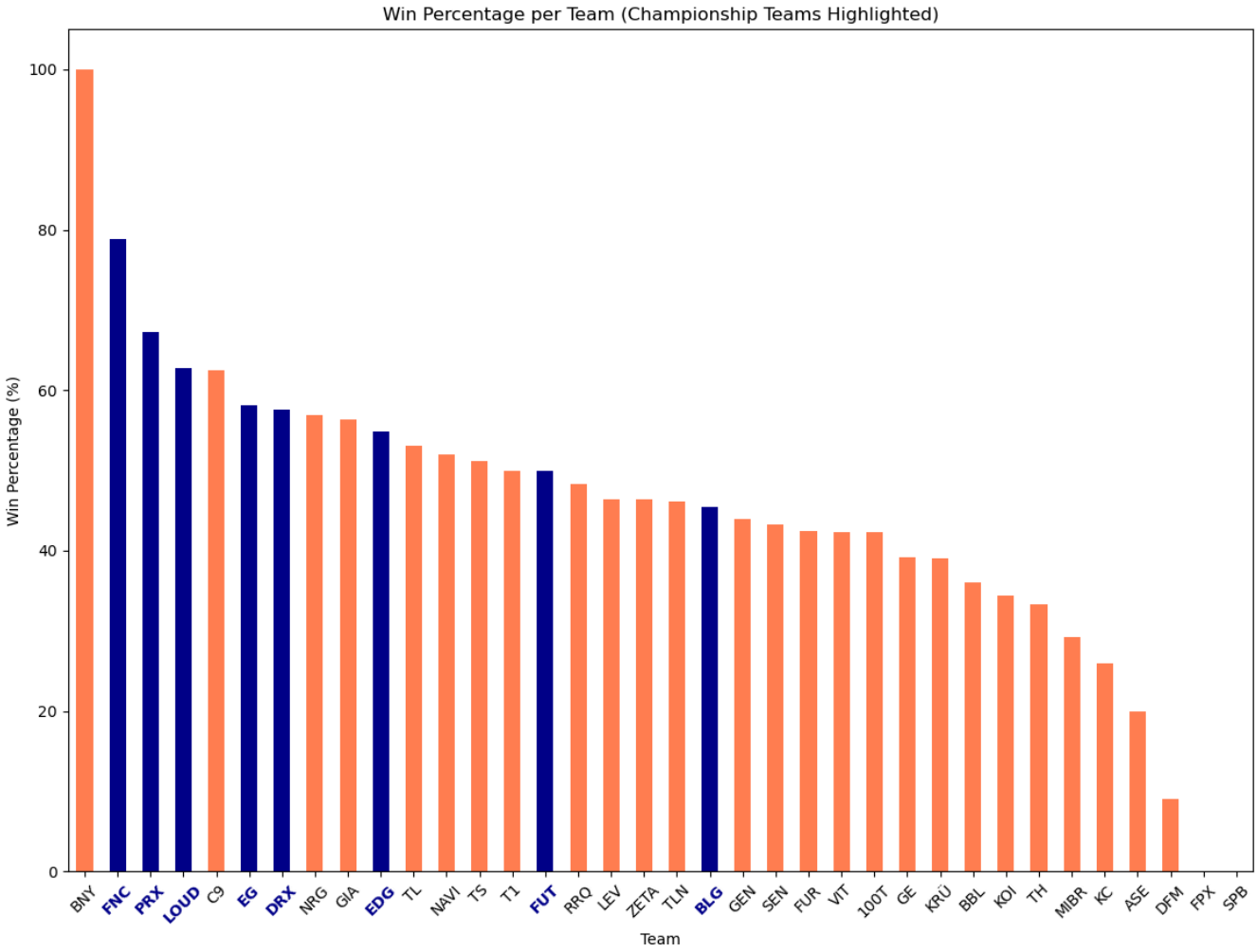
Data Wrangling: This was the most simplest since the code was well formatted and developed. There were no missing values which makes the data more readable to our code. Additionally, outliers are encouraged since it highlights excellent team work which is predicated on the agents they play. Feature Engineering will be highlighted with the analysis of the models.

Exploratory Data Analysis: Despite all of the EDA not being necessary for the predictive modeling and deep learning classes it still provides valuable insight for the clustering done in the recommendation systems class. First, let's start with the correlation matrix and how that influenced the feature engineering of the traditional models.



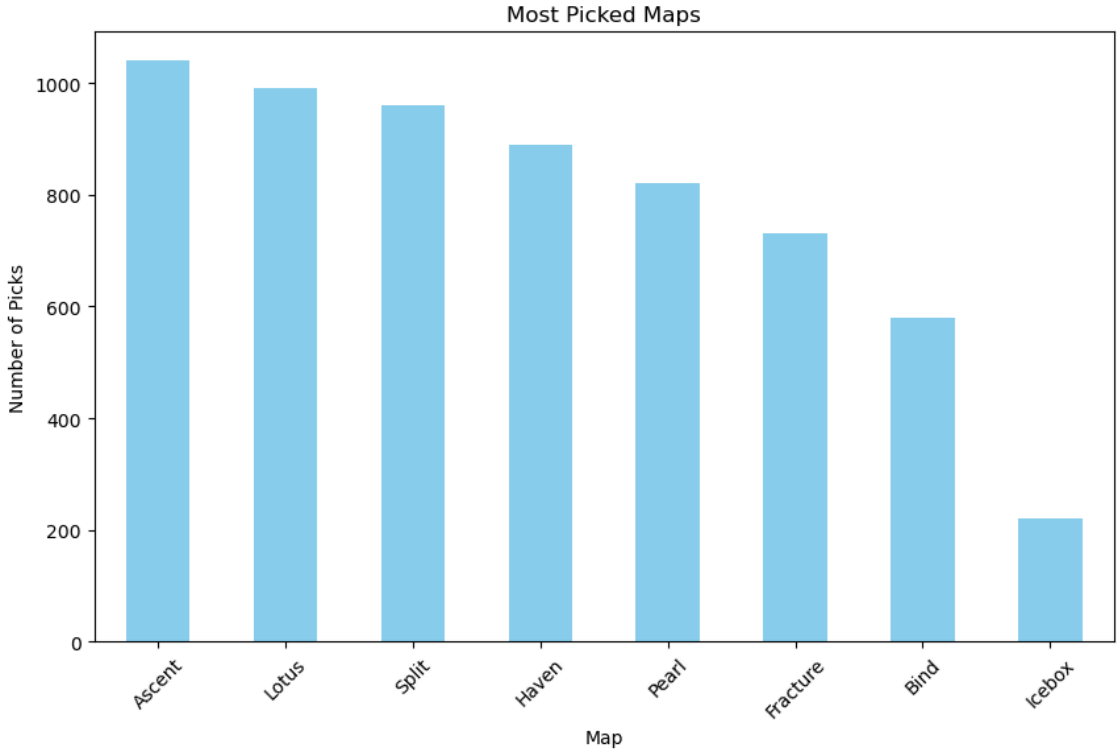
The correlation matrix indicates that ACS, ADR (average damage per round) and # kills would all be good features to use for our prediction, since the correlation between them is strong and higher values of these statistics indicate good player performance, these are likely reliable features. Later during the modeling section there is some analysis into which features are the most important.

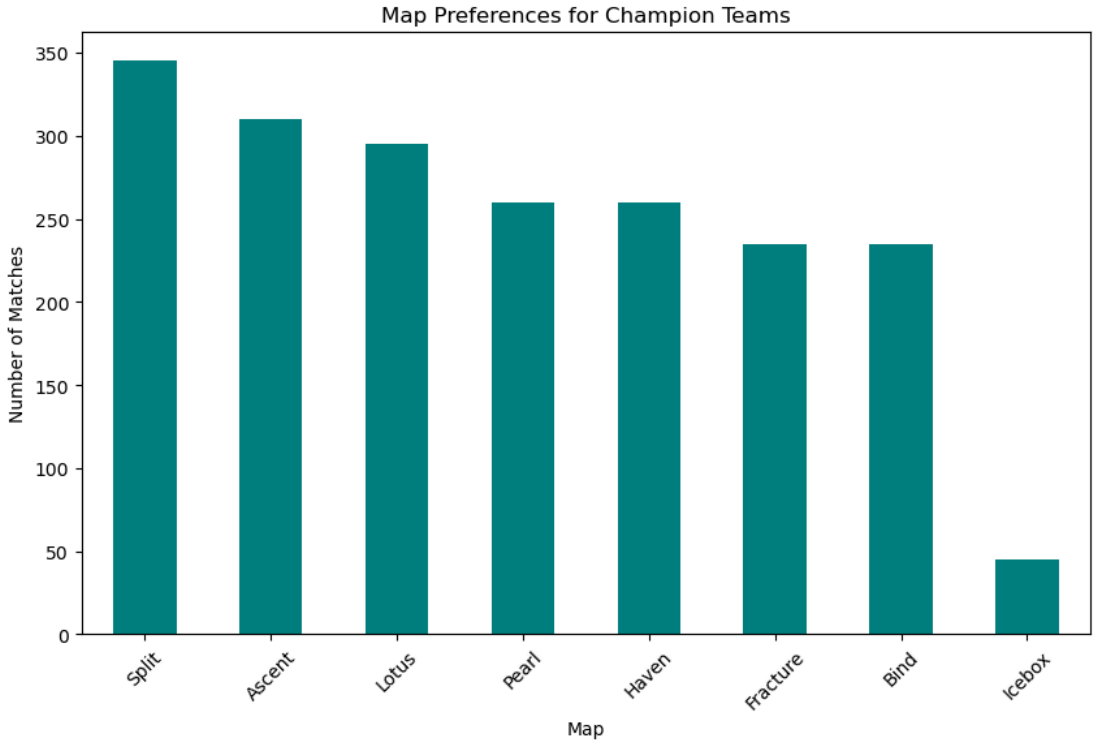
Certain plots that were visualized show which parts of our cluster would be most reliable:



This plot is meant to indicate the champion teams highlighted in blue. A certain point of contention was if only those teams should be used because of better performance overall. But, what can be seen from the plot is that there were many teams who rivaled their win percentages, and only analyzing 10 teams when there are many more might not be a good indicator of adaptable strategies. So clustering or modeling using this data would not be very encompassing.

These next two plots show the difference in map picks between all the teams and just the champion teams.

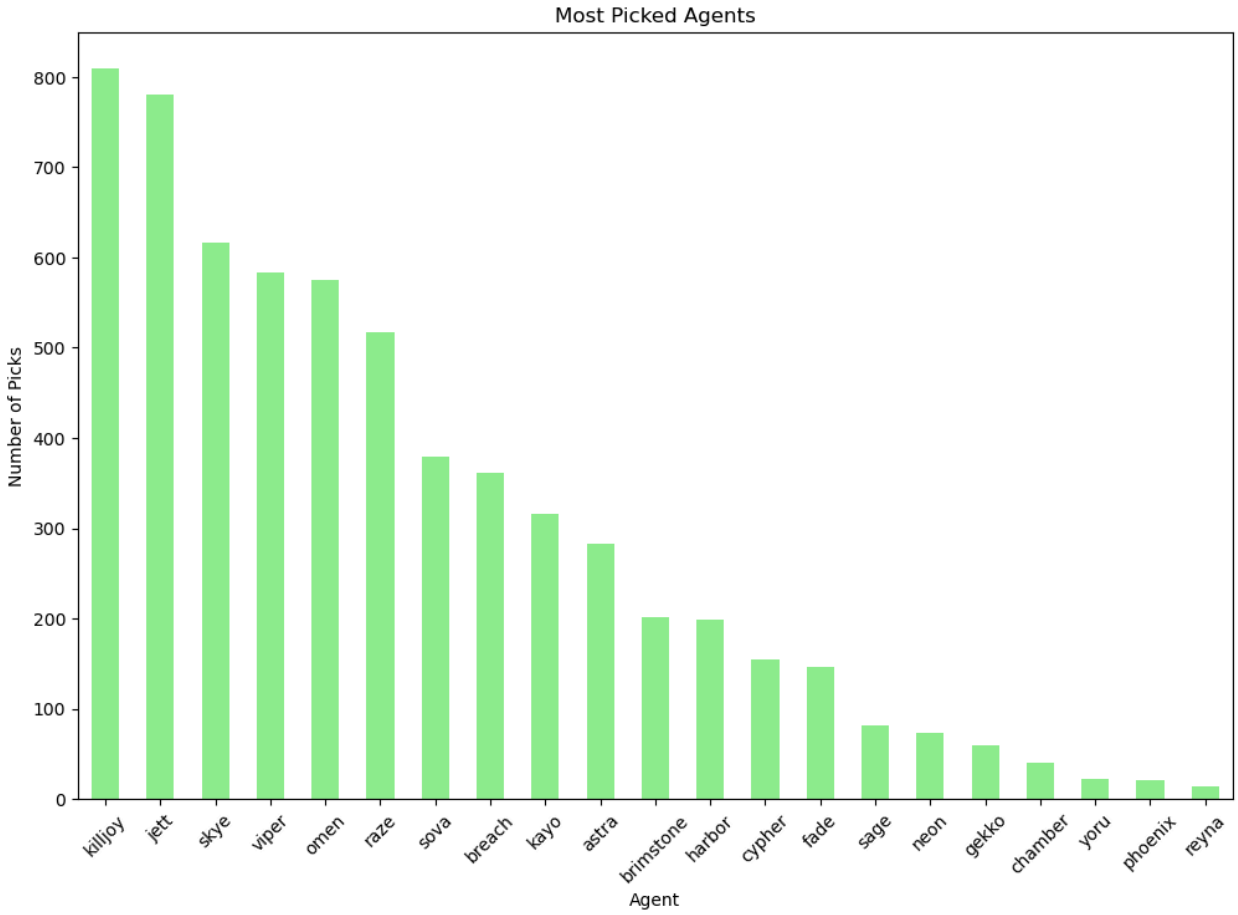




This first plot indicates that the map ‘Ascent’ was chosen almost 800 times over ‘Icebox’, this means the cluster for the most chosen map would have more data to use over others.

The reason why the champion teams needed to be plotted is to show how little data they give us and why it should not be used for the modeling. Some may contend that because these teams were the best of the best the results would be of higher quality, but that is not true. The map ‘Icebox’ would only have 50 rows of data to pull from, which is a really small size compared to the most chosen champions map ‘Split’ which was chosen more times then ‘Icebox’ was using the bigger data set.

The next plot shows the most chosen agents for all the teams, and this will highlight a problem our clustering model will have:



The problem with this plot is that same agents like Reyna, Phoenix, or Yoru have very low pick rates, meaning the data surrounding their use would be quite unreliable. But they probably will not be shown frequently in our cluster and not with a high yield.

Modeling:

Traditional Predictive Modeling: In order to do predictive analysis, this project decided to compare traditional predictive modeling to deep learning to make some notable insights about the game and data.

This goal is to find out if particular features have the ability to accurately predict whether or not one will win a valorant game.

This is a **binary classification problem**, Since the target variable is being encoded as 1 for a 'team win' and 0 otherwise, this is a binary classification problem where the model will predict two distinct classes (win or loss)

Data Preprocessing

- Data Cleaning: Non-numeric characters are removed from numerical columns, ensuring the data is in the correct format for modeling.
- Feature Encoding: Categorical features are encoded using OneHotEncoder to convert them into numerical values without imposing ordinality/
- Data Scaling: The StandardScaler is applied to numerical features to standardize them, which is particularly beneficial for models that are sensitive to the scale of the data, such as logistic regression.
- Missing Value Handling: SimpleImputer is used to fill missing values, preserving the integrity of the dataset for training.

.

Models:

1. **Random Forests --** less likely to overfit, it can handle a large number of features which is good for testing feature importance, and its nonlinear which is good for complex non-linear interactions, like video games.
2. **Gradient Boosting Classier** -- supposedly high accuracy since it corrects errors and gradient boosting makes good use of hyperparameters.
3. **Logistic Regression** -- good baselines model for binary classification, verify ast and simple, if high accuracy then the data is not complex.

Hyperparameter Tuning:

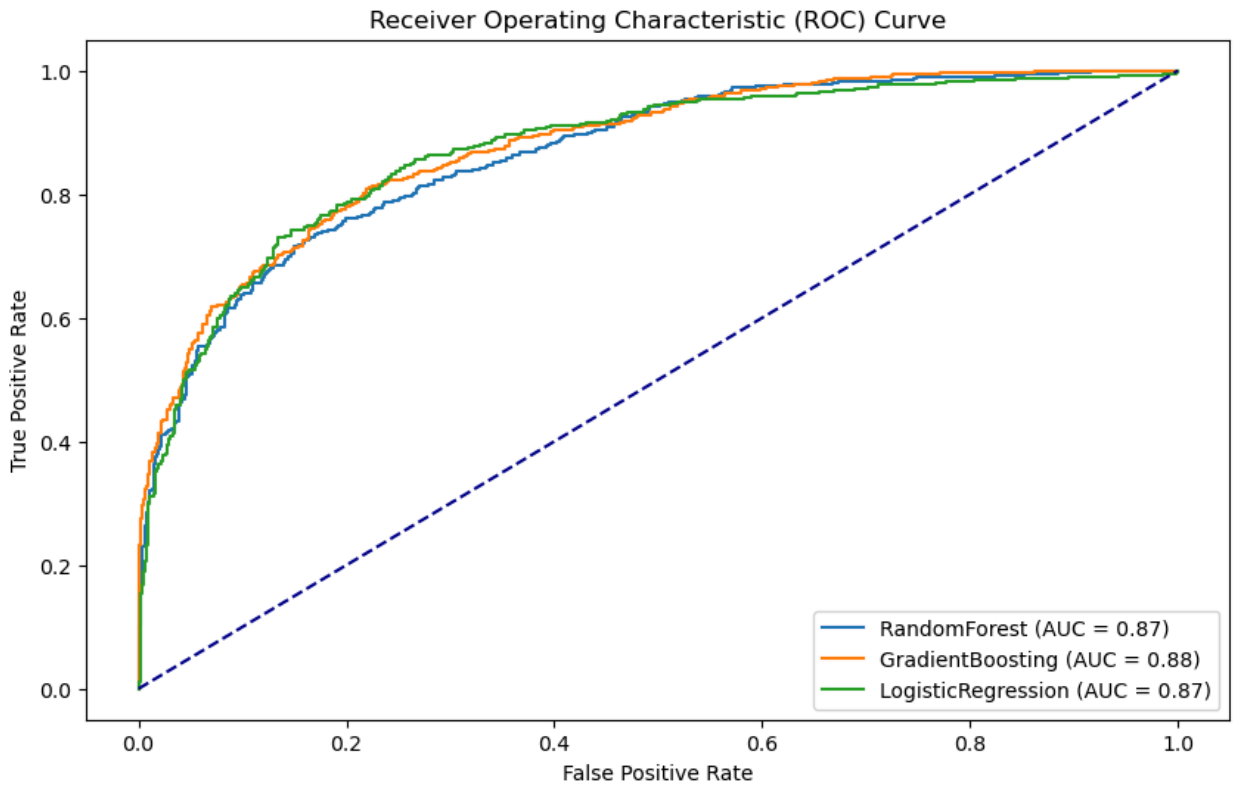
- RandomizedSearchCV: It is used over GridSearchCV for efficiency, sampling a random subset of hyperparameters within predefined ranges to find an optimal combination for model performance.
- Custom Hyperparameters: Specific hyperparameters are chosen based on model characteristics and domain knowledge, such as the depth of trees in ensemble methods and the regularization strength in logistic regression.

Training:

- Pipeline: A pipeline structure is used to streamline preprocessing and modeling steps, ensuring consistency and reproducibility.
- Cross-validation: The training process incorporates cross-validation, providing a more robust evaluation by using multiple splits of the data.

Evaluation Metrics:

- **Accuracy**: All three models show similar accuracy (around 0.79), indicating that approximately 79% of predictions are correct. This suggests that the models have a good overall rate of correct predictions for this dataset.
- **Precision**: The precision is also at 0.79 for LogisticRegression, which means that when the model predicts a team will win, it is correct about 79% of the time. This is valuable in scenarios where the cost of false positives (predicting a win when it's a loss) is high.
- **Recall**: With a recall of 0.79, the models are able to capture 79% of the actual wins. This is important in scenarios where missing out on true wins (false negatives) is costly.
- **ROC AUC:** The GradientBoosting model has the highest ROC AUC of 0.88, indicating a very good ability to distinguish between the winning and losing classes. ROC AUC is a robust metric as it evaluates model performance across all classification thresholds.



The accuracy, precision, recall, f1-score, and auc-roc score are all very similar. This is due to both classes being equally balanced and the features being very impactful for all models.

Feature Importance:

- The feature importances from the RandomForest and GradientBoosting models give us insight into what factors are most predictive of a win in Valorant. Notably, 'death' is the most significant feature for both models, suggesting that the frequency of player deaths is a strong indicator of the match outcome.
- Other important features like 'kill', 'assist', 'acs', and 'adr' are also indicative of a player's contribution to the team's success, the three features mentioned earlier play a heavy role.

Deep Learning:

Data Preprocessing:

- Preprocessing steps include cleaning numerical features, handling missing values, scaling numerical features for normalization, and encoding categorical features.
- The preprocessing method returns scaled and encoded feature arrays ready for model input, along with a binary-encoded target variable array derived from a 'win_lose' column.

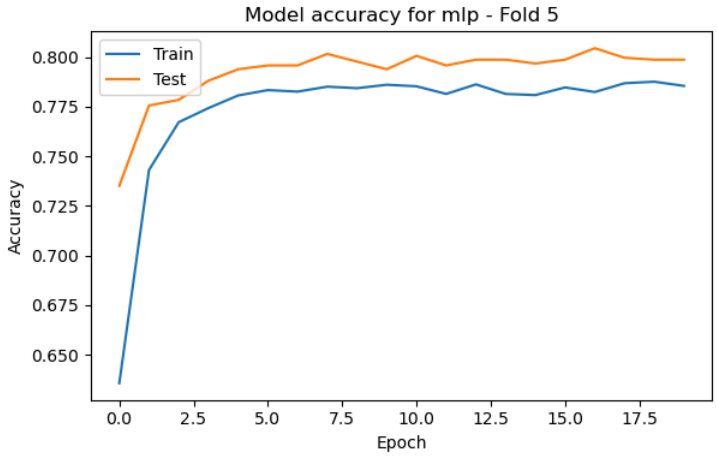
Models:

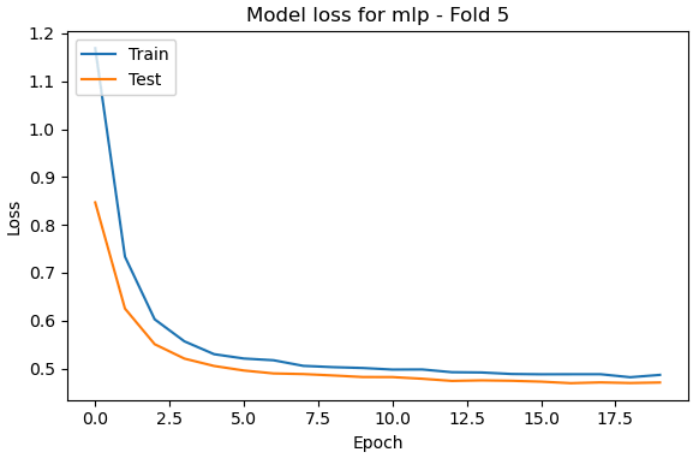
- Multi-Layer Perceptron (MLP): A basic dense neural network suitable for tabular data.
- Long Short-Term Memory (LSTM): A type of recurrent neural network (RNN) optimized for learning order dependence in sequence prediction problems.
- Simple RNN: A more straightforward RNN architecture for learning sequential patterns.
- Convolutional Neural Network (CNN): Primarily known for image processing but adapted here for time-series data.

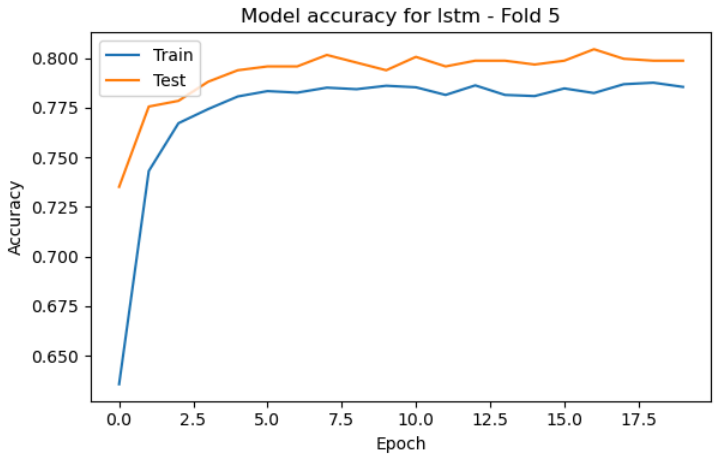
Training:

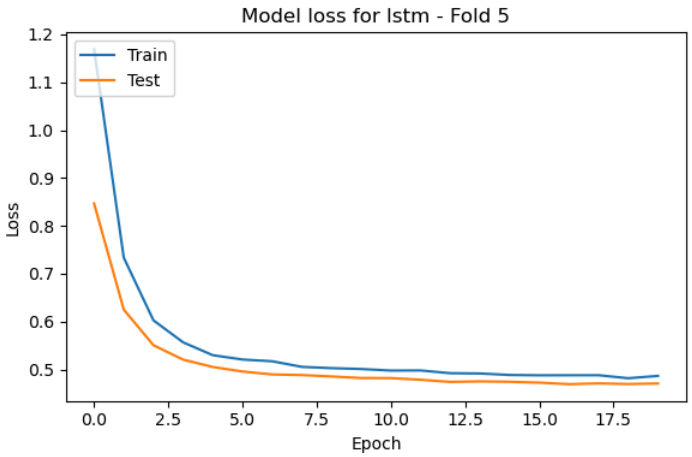
- The class includes a generalized cross-validation method tailored for time-series data (TimeSeriesSplit), demonstrating the use of LSTM, Simple RNN, and CNN models for sequential data prediction. This reflects an understanding of the importance of temporal dynamics in the dataset.
- Models are compiled with Adam optimizer and binary cross-entropy loss function, reflecting a binary classification task. Precision and recall metrics are tracked, emphasizing the importance of both positive prediction accuracy and coverage.
- Early stopping is employed to prevent overfitting by halting training when validation loss ceases to decrease, showcasing an awareness of the challenges in training deep learning models.

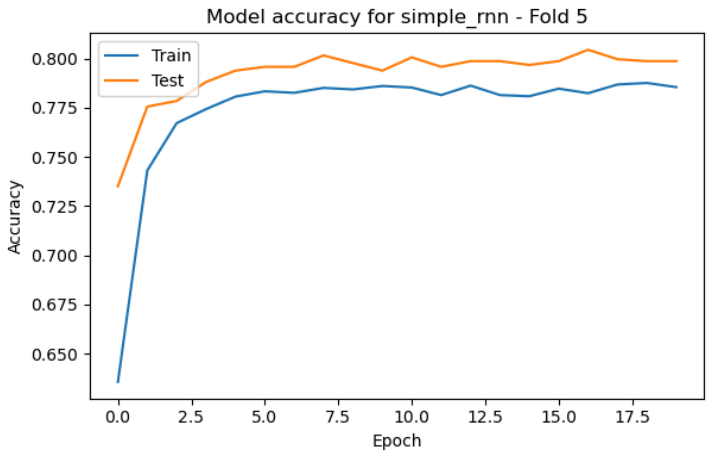
Evaluation and Visualization:

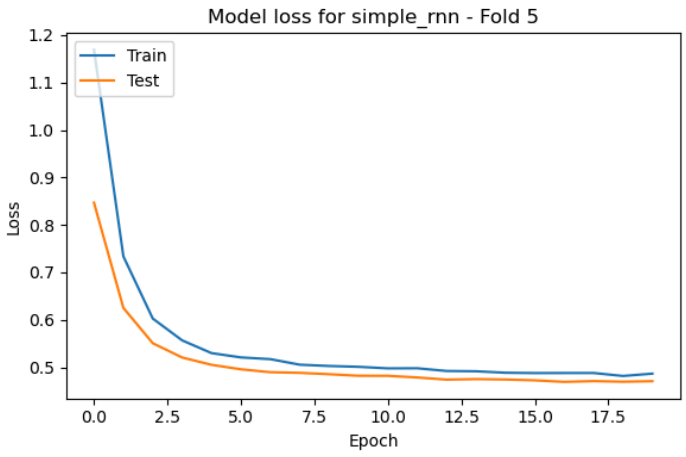


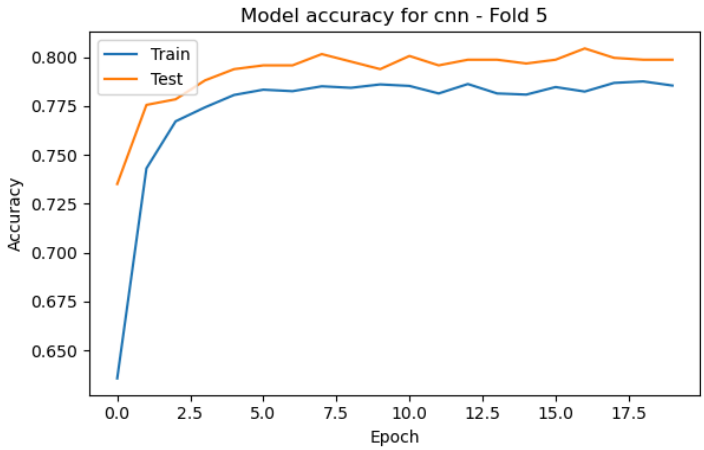


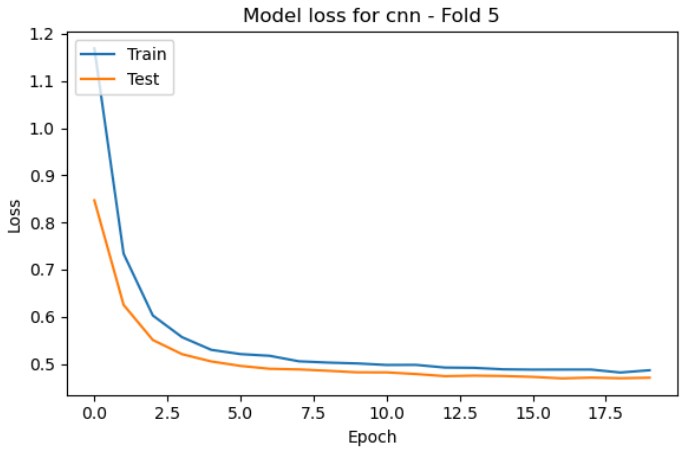








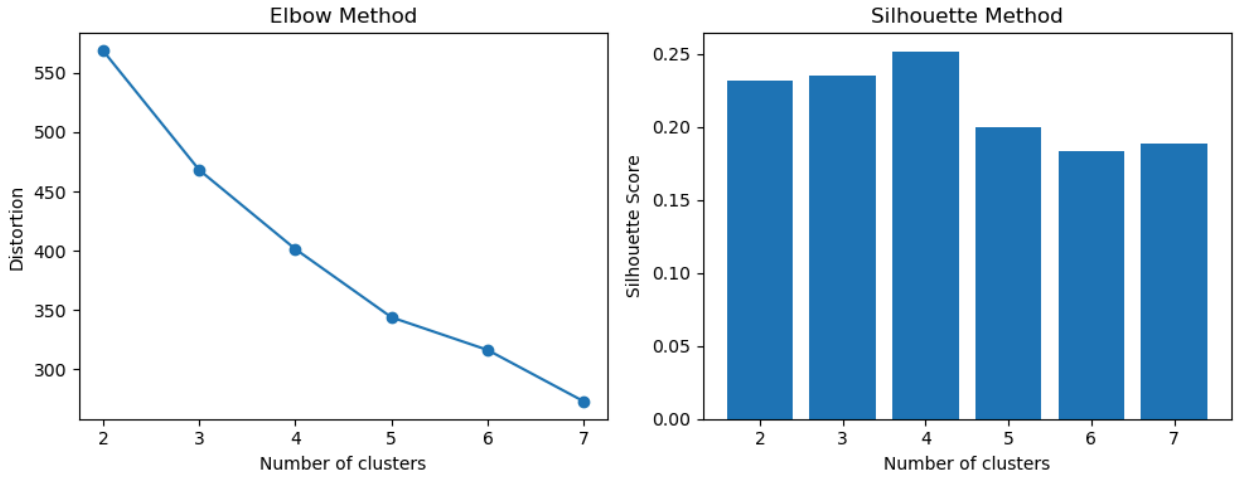




Clustering for Recommendation Systems:

Non-clustering calculation: A method called agent_performance calculates the average performance metrics for agents on each map, displaying the top-performing agents. This could help teams understand which agents are typically most effective on any given map if you wanted to play them individually.

K-Means Clustering: The cluster_teams method applies KMeans clustering to group similar team compositions together. A 3D visualization of the clusters is provided, using PCA for dimensionality reduction.



- **How many Clusters:** The elbow method looks for a point of distortion where decreases happen slowly, this seems to be at cluster 3 or 4. The Silhouette Method measures how similar an object is to its own cluster compared to others. Cluster 2, 3, and 4 show a higher silhouette score, but there is a sharp decline after. **Three clusters** ended up being

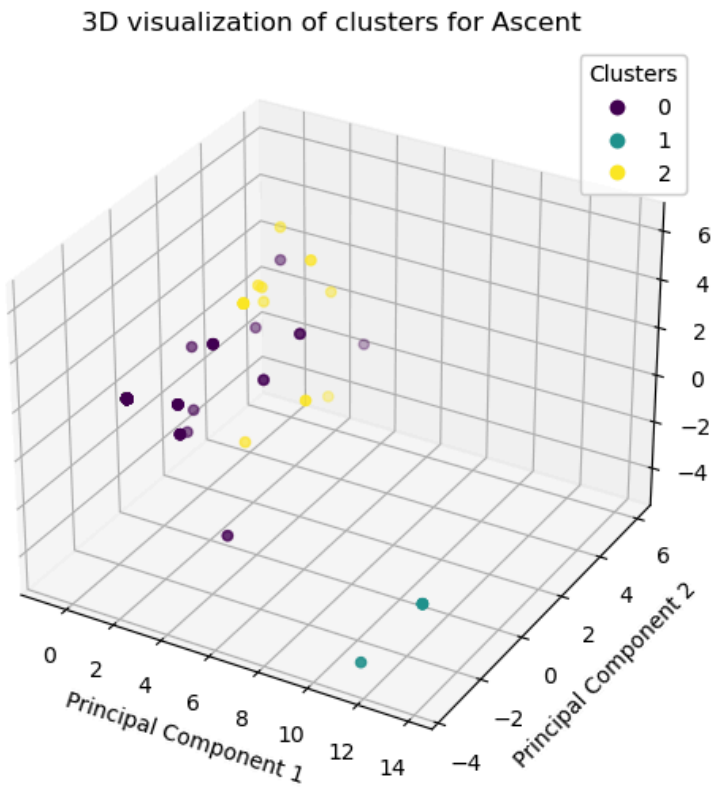
used because the elbow in the plot is a little more pronounced and the silhouette score difference was next to marginal.

- **Standardization:** The dummy variables for each agent are standardized to ensure that each agent has the same weight in the Euclidean distance calculations used by the K-means algorithm. In Valorant, this step ensures that no single agent's presence will disproportionately influence the clustering due to scale differences.
- **The K-means algorithm** is applied to the standardized data to find clusters of similar team compositions. In Valorant, this would identify common strategies or team setups used across different matches.
- **PCA Analysis** is a technique that is often used alongside K-Means Clustering. It is used to visualize the patterns in a reduced dimensionality space like clustering. The axes do not represent a single agent, but a combination of agents.

Cluster Results:

Using one map ‘Ascent’ as an example

Cluster 0:	Cluster 1:	Cluster 2:
kayo 0.994737	cluster 1.00	cluster 2.000000
omen 0.989474	viper 1.00	omen 0.928571
jett 0.989474	harbor 1.00	killjoy 0.857143
sova 0.889474	reyna 1.00	jett 0.785714
killjoy 0.836842	raze 1.00	skye 0.642857
fade 0.094737	skye 0.75	viper 0.642857
cypher 0.078947	kayo 0.25	fade 0.285714
viper 0.052632	phoenix 0.00	breach 0.285714
sage 0.031579	sova 0.00	phoenix 0.142857
raze 0.015789	sage 0.00	cypher 0.071429
astra 0.010526	astra 0.00	harbor 0.071429
gekko 0.010526	breach 0.00	kayo 0.071429
skye 0.005263	killjoy 0.00	raze 0.071429
phoenix 0.000000	jett 0.00	sova 0.071429
reyna 0.000000	gekko 0.00	astra 0.071429
breach 0.000000	fade 0.00	gekko 0.000000
harbor 0.000000	cypher 0.00	reyna 0.000000
	omen 0.00	sage 0.000000



Left value is the agent or character you can play as, the right value represents the percentage it that agent is present in a team composition.

For cluster 0 (purple): The first 5 agents clearly dominate, indicating that these 5 are probably often used together, making this team composition of 5 optimal and recommended. With the other agents used very sparsely in lieu of one of top 5 chosen agents.

For cluster 1 (blue): Only 6 are used, and 4 of which are used all the time, indicating that these clusters of compositions are niche and rare. As you can see in the plot they are far away from the other clusters.

For Cluster 3 (yellow): Due to the differences in percentage used having a slightly greater difference than cluster one and it being more spread out, this cluster is more flexible in what you could use together. Combines previous clusters opting for a more hybrid approach as you can see due to the results.

How can this be used:

Agent Recommendations: Based on the prevalent compositions within a cluster, a recommendation system can suggest agents to players that complement the existing picks. For example, if a team has selected agents that are commonly found in Cluster 0 but hasn't chosen Kayo, the system might recommend picking Kayo to align with successful team strategies.\

Strategic Insights: For new or less experienced players, the system can provide insights on which agents are often picked together, helping them understand synergistic relationships and popular strategies within the game.

Future Improvements:

Personalized Suggestions: By analyzing a player's history and identifying which cluster their gameplay most often aligns with, the system can make personalized agent recommendations that fit their playstyle.

Algorithm Enhancements:

- Utilize More Advanced Clustering Techniques: Explore methods like DBSCAN or HDBSCAN that can identify clusters of arbitrary shape and are better at handling noise and outliers.
- Incorporate Time Series Analysis: Consider the temporal aspect of data, as team compositions and strategies may evolve over time.

Model Testing:

- **A/B Testing**: Run experiments to compare different recommendation models and determine which yields the most successful outcomes.
- **Post-Match Analysis**: Compare the recommended team compositions with actual match results to validate and adjust the recommendation logic.