**Artifact 3 – Databases**

**Artifact Description**

This was my first machine learning project, and it really shows. I found a lot of sloppy code to correct and organize while adding my enhancement. This project is centered around using traditional machine learning techniques to create a model which predicts whether someone died in the Titanic disaster based on passenger information. This included things like the passenger’s sex, age, ticket fare, name, etc. In the original project, the same models were trained but there was no use of a database and the notebook was much less organized.

**Artifact Justification**

I selected this item as it shows mastery of traditional machine learning skills. Things like random forests, gradient boosted algorithms, and KNN models are still very commonly used and sometimes are the best tool for the task. I wanted to show future employers my ability to use and improve these models on a dataset. The use of these various model architectures shows my skills in engineering a wide array of machine learning solutions.

The enhancement improved this artifact by showing my ability to store and query data in an external database. My enhancement involved the creation a NoSQL database using MongoDB. In this database the name of the model and its weights were stored. Afterwards the database was queried to instantiate a new version of the model using the results of the query. Doing this showed not only that the model was stored successfully, but that it could be retrieved from outside of the notebook itself for use in other areas. This highlights the ability to deploy trained models in other applications, a highly valuable data science skill.

**List of Improvements by Section**

1. Features

* Fixed typos in chart which defined sibsp and parch incorrectly.

1. Data Exploration

* Implemented read\_csv\_with\_check function to validate existence of CSV files before creating dataframes.
* Eliminated importation of the competition’s testing dataframe in favor of creating my own testing frame from the training data. The competition’s training data does not have answers and accuracy of it cannot be measured in the notebook. Creating my own allows me to have a better understanding of model performance on new data.

1. Filling Data and Converting All Data to Integers

* Eliminated name, ticket number, and cabin features from dataset for reasons explained in notebook. This resulted in much faster training times and more accurate models.
* Changed the transformation of the feature describing the passenger’s sex to use one hot encoding rather than label encoding. This eliminated unintended implied ordinality that results from label encoding and created more accurate models.
* Transformed all data to float32 datatype. This is a standard commonly used in machine learning models which helps in the batching and training processes that I was unaware of when first creating this notebook.

1. Creating a Baseline Score

* Moved this section to come before training any other models. Makes the progression of the notebook much more logical to others reading it.

1. Training models

* In original notebook, training with various model architectures was scattered throughout the notebook. Placed them all here to improve organization.

1. Hypertuning with GridSearchCV

* My various changes in the data transformation portion made the Gradient Boosted model outperform the Random Forest Classifier unlike in the original notebook. GridSearch was done for this instead.

1. Evaluating Predictions of the best performing model

* Adjusted the Feature Importance graph to be organized from greatest to least in horizontal format, makes it more readable and understandable.

1. Measuring Cross Validated Metrics

* Cleaned up repeated printing of accuracy metrics in favor of a new confusion matrix.

1. Exporting the Models (new, planned database enhancement)

* Created a MongoDB database to house models.
* Saved to database from notebook while maintaining security by hiding username and password needed to access the database behind environmental variables.
* Demonstrated loading of model is equal in performance to locally trained model.

**Reflection**

The creation of this model reinforced my traditional machine learning, data visualization, and data preparation skills immensely. During improvement, I learned a lot about improving old work and using MongoDB. This was my first machine learning project, and I made a lot of rookie mistakes when I first built it. For example, when preparing the data frame to train the model I label encoded each name of each passenger. This resulted in the creation of a complex column where each value implied some sort of ordinance. The best practice really would have been to drop the column all together, as each name value is unique and would not produce any sort of pattern that an AI can learn. This realization is something I learned in the improvement process and helped me to feel more confident in my data processing skills.

The biggest challenge in implementing the enhancement was making it so the database could be queried from the notebook itself. Many databases only allow certain IP addresses to access them at all and will block any IP not whitelisted. The problem was that a Kaggle notebook does not use a steady IP address and runs on Kaggle’s servers. I solved this by allowing any IP to access the database, but only allowing them to do anything with it if proper credentials are given. These credentials were stored as secret environmental variables only known by me, with the result being I could access the database from any IP address but only I could make changes or query it since only I knew the credentials.

I did meet the course objectives I had planned for in this project. The objective I strived to meet was, “Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources”. I did this through the implementation of a MongoDB database to house and load the models from. This was done while using environmental variables to obscure the mechanism used for requests. This allows me to use the database in a secure way while not exposing how to access it publicly. Should the method to access the database be exposed publicly, it opens an application to data leaks, wiping of the database, and tremendous expenses. To avoid these outcomes, I made sure that the database could only be accessed in a secure way.