**Machine Learning Pipeline in Quartz**

**Introduction to a problem:**

In the current Quartz application, we don’t have any python package to develop a machine learning model and deploy the model in a particular environment.

**Proposed Solution:**

Identifying the common steps and requirements across different machine learning projects in the Bank. Such as data preparation, preprocessing, feature engineering, model training and evaluation metrics. To create a standard template for the Machine Learning pipeline that includes the common steps and requirements as a python class or set of python functions in Quartz that can be called sequentially to run the pipeline. Customizing the preprocessing, feature engineering, and model training templates to fit the specific requirements of each Machine Learning project can be re-used in Bank of America. This pipeline includes modifying the data preprocessing and feature engineering steps to fit the data and problem domain, selecting suitable model architectures and hyperparameters, and defining evaluation metrics that are relevant to use-case specific.

Customize options and proper training and support to teams in the bank on how to use the machine learning pipeline, including how to modify the template for their specific project requirements. Also, Quartz establishes version control and documentation practices to ensure that the machine learning pipeline is maintained and updated as needed and that changes are tracked.

1. **QzTable with data**

A QzTable with raw financial data or real-time bank data can provide valuable information for financial analysis, modeling, and decision-making.

1. **Data transformation and Cleaning**

Data cleaning is a crucial step in data transformation, as it involves identifying and correcting errors, inconsistencies, and inaccuracies in the data. Techniques like filling in missing values, removing duplicates, and correcting typos can help clean the data and make it more suitable for machine learning. Filling missing values is particularly important, as missing data can cause bias in machine learning models. Removing duplicates can also improve the quality of the data and reduce the risk of overfitting. Finally, correcting typos can help improve the accuracy of text-based machine learning models, which are often prone to errors due to misspellings and other inconsistencies in the data. Overall, data cleaning and transformation techniques are essential in preparing data for machine learning and can significantly impact the performance and accuracy of machine learning models.

1. **Data Engineering and Selection**

Data engineering and feature selection are critical steps in building accurate machine-learning models. Data engineering involves creating new features or transforming existing ones to better represent the underlying patterns and relationships in the data. Feature selection, on the other hand, involves identifying the most relevant features for the task at hand, and discarding the ones that are irrelevant or redundant. Common data engineering techniques include one-hot encoding, normalization, and scaling, while feature selection techniques include correlation-based methods, wrapper methods, and embedded methods. One-hot encoding is used to represent categorical features as binary values, while normalization and scaling can help improve the performance of models that are sensitive to the scale of the features. Correlation-based methods evaluate the relationship between each feature and the target variable, while wrapper methods and embedded methods use machine learning algorithms to determine the most important features. Overall, data engineering and feature selection are critical steps in building accurate and efficient machine learning models, and can significantly improve the performance and interpretability of the models.

1. **Splitting training and test data**

Splitting the data into training and test sets is an essential step in building machine learning models. The training set is used to train the model, while the test set is used to evaluate the performance of the model on unseen data. Splitting the data into training and test sets helps to prevent overfitting, which occurs when the model memorizes the training data and performs poorly on new, unseen data. Generally, the data is split into a ratio of 70:30 or 80:20 for training and testing, respectively. However, the appropriate ratio may depend on the size of the dataset and the complexity of the model. It is important to ensure that the training and test sets are representative of the underlying distribution of the data to ensure that the model generalizes well to new, unseen data. Cross-validation is another technique that can be used to evaluate the performance of the model on different splits of the data and ensure that the model is robust.

1. **Building different ML models**

Building different regression or classification models on top of the same dataset can help to identify the best algorithm for the given task and improve the overall performance of the models. There are various regression and classification algorithms available, such as linear regression, logistic regression, decision trees, random forests, support vector machines, and neural networks, among others. Each algorithm has its own strengths and weaknesses, and may perform differently depending on the characteristics of the dataset and the task at hand. By building multiple models and comparing their performance, it is possible to identify the algorithm that works best for the given task and potentially improve the accuracy and generalization of the models. It is important to ensure that the models are trained and evaluated using the same data splits and evaluation metrics to make a fair comparison. Additionally, it is crucial to interpret the results and understand the strengths and limitations of each algorithm in order to make informed decisions and recommendations based on the models' performance.

**Regression algorithms:**

Linear Regression

Ridge Regression

Lasso Regression

Elastic Net

Least Angle Regression (LARS)

LARS Lasso

Orthogonal Matching Pursuit (OMP)

Bayesian Ridge

Automatic Relevance Determination (ARD)

Passive Aggressive Regressor

TheilSen Regressor

Huber Regressor

Kernel Ridge

Support Vector Regression

Decision Tree Regressor

Random Forest Regressor

Extra Trees Regressor

AdaBoost Regressor

Gradient Boosting Regressor

Extreme Gradient Boosting (XGBoost) Regressor

Light Gradient Boosting Machine (LightGBM) Regressor

CatBoost Regressor

Neural Network (Multi-layer Perceptron) Regressor

K-Nearest Neighbors (KNN) Regressor

Stochastic Gradient Descent Regressor

Classification algorithms:

**Logistic Regression:**

K-Nearest Neighbors (KNN) Classifier

Naive Bayes

Decision Tree Classifier

Random Forest Classifier

Extra Trees Classifier

Gradient Boosting Classifier

Extreme Gradient Boosting (XGBoost) Classifier

Light Gradient Boosting Machine (LightGBM) Classifier

CatBoost Classifier

Linear Discriminant Analysis (LDA)

Quadratic Discriminant Analysis (QDA)

Neural Network (Multi-layer Perceptron) Classifier

AdaBoost Classifier

Linear Support Vector Classification (Linear SVC)

Kernel Support Vector Classification (Kernel SVC)

Radius Neighbors Classifier

Stochastic Gradient Descent Classifier

Gaussian Process Classifier

1. **Evaluate models**

Evaluating models is a critical step in building machine learning models and involves measuring the performance of the model on unseen data. There are various metrics that can be used to evaluate the performance of regression and classification models, such as mean squared error, R-squared, accuracy, precision, recall, F1 score, and ROC AUC score, among others. The appropriate metric(s) may depend on the characteristics of the dataset and the task at hand. It is important to ensure that the model is evaluated using the appropriate metrics and that the results are interpreted in the context of the problem being solved. Additionally, it is crucial to avoid overfitting and ensure that the model generalizes well to new, unseen data. Cross-validation and hyperparameter tuning are techniques that can be used to evaluate and optimize the performance of the model. Overall, evaluating models is a key step in building accurate and robust machine learning models and can help to inform decision-making and provide insights into the underlying patterns and relationships in the data.

1. **Train the best model with fine-tuning**

Training the best model with fine-tuning involves optimizing the hyperparameters of the model to improve its performance on the task at hand. Hyperparameters are the parameters that are set before training the model and controlling its behavior, such as learning rate, regularization strength, and the number of hidden layers. Fine-tuning involves a systematic search of the hyperparameter space to find the best combination of hyperparameters that results in the best performance of the model on the validation set. This process may involve using techniques such as grid search, random search, or Bayesian optimization to search for the best hyperparameters. Once the best hyperparameters have been identified, the model is trained on the entire training set and evaluated on the test set to ensure that it generalizes well to new, unseen data. Fine-tuning can help to improve the performance of the model and make it more robust to different variations in the data. However, it is important to ensure that the model is not overfitting and that the results are interpreted in the context of the task being solved.

1. **Save and Deploy model**

Saving and deploying a machine learning model involves the process of persisting the model to a file or storage location, and then making it available for use in a production environment. Once the model is trained and evaluated, it can be saved in a standardized format such as the Hierarchical Data Format (HDF5) or the Protocol Buffers format (protobufs) to preserve the architecture, weights, and hyperparameters of the model. The saved model can then be loaded and used for prediction on new, unseen data. Deploying the model involves making it available as an API or a web service, which can be used by other applications or systems to make predictions in real time. It is important to ensure that the deployed model is secure, efficient, and scalable to handle large volumes of requests. Overall, saving and deploying a machine learning model is a crucial step in applying it to real-world problems and can have a significant impact on business outcomes and decision-making.

**Business benefit:**

* ML Pipeline can be used by different business units across the Bank of America.
* Improved Efficiency: A predefined pipeline can help automate the process of building machine learning models, which can save time and resources. This can be especially important in situations where data scientists or analysts are working on multiple projects simultaneously.
* Consistency and Standardization: A predefined pipeline ensures that the same set of steps is followed every time a model is built, which can help to standardize the process and ensure consistency in the models. This can help to improve the reproducibility and reliability of the models.
* Improved Accuracy: A predefined pipeline can help to identify the best algorithm and hyperparameters for the given task, which can improve the accuracy and generalization of the models. This can result in better predictions and decision-making.
* Faster Deployment: A predefined pipeline can help to speed up the process of model deployment, as the model is already trained and evaluated. This can help to reduce the time to market and increase the agility of the business.
* Better Insights: A predefined pipeline can help to provide better insights into the underlying patterns and relationships in the data, which can inform business decisions and strategy. This can be especially important in situations where there is a large volume of data and complex relationships between the variables.

**Conclusion:**

In summary, building a machine learning pipeline for classification and regression models can provide several benefits to businesses, including improved efficiency, consistency, accuracy, faster deployment, and better insights. In the context of Bank of America, creating a standardized machine learning pipeline that can be reused by different teams can save time and effort by reducing the need to reinvent the wheel for each project, and ensuring the best practices and common requirements are consistently applied across different projects. This can help to improve the overall effectiveness and impact of machine learning projects in the organization.

**References**

* Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, Inc.
* Raschka, S., & Mirjalili, V. (2020). Python machine learning: machine learning and deep learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing Ltd.
* Oliphant, T. E. (2015). A guide to NumPy. USA: Trelgol Publishing.
* Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.