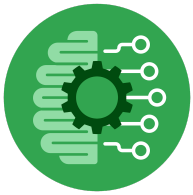
Course Six
The Nuts and Bolts of Machine Learning



Instructions

Use this PACE strategy document to record decisions and reflections as you work through the end-of-course project. As a reminder, this document is a resource that you can reference in the future and a guide to help consider responses and reflections posed at various points throughout projects.

Course Project Recap

Regardless of which track you have chosen to complete, your goals for this project are:

- [ ] Complete the questions in the Course 6 PACE strategy document
- [ ] Answer the questions in the Jupyter notebook project file
- [ ] Build a machine learning model
- [ ] Create an executive summary for team members and other stakeholders

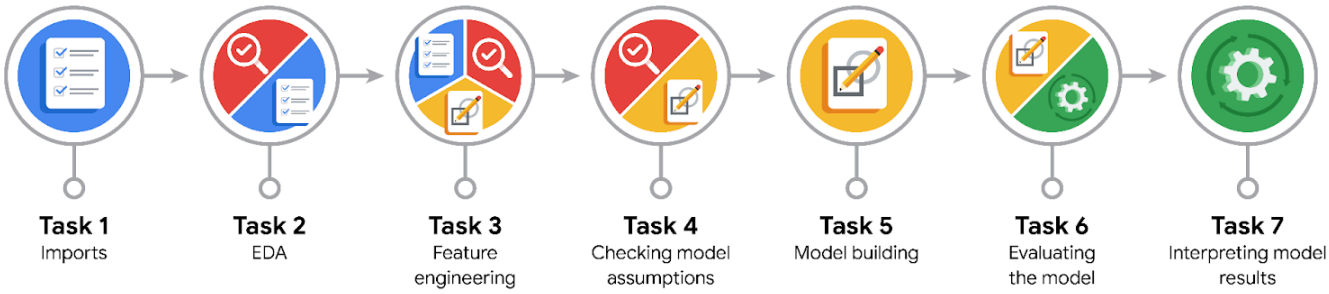
Relevant Interview Questions

Completing the end-of-course project will empower you to respond to the following interview topics:

- What kinds of business problems would be best addressed by supervised learning models?
- What requirements are needed to create effective supervised learning models?
- What does machine learning mean to you?
- How would you explain what machine learning algorithms do to a teammate who is new to the concept?
- How does gradient boosting work?

Reference Guide:

This project has seven tasks; the visual below identifies how the stages of PACE are incorporated across those tasks.



Data Project Questions & Considerations

PACE: Plan Stage

- What are you trying to solve or accomplish?

Initial Client Request: Automatidata's client, the NYC TLC, has requested a machine learning model to predict if a customer will not leave a tip. The stated goal is to use this model in an app to alert drivers, ostensibly to help them increase revenue.

Ethical Re-evaluation: As identified in the initial analysis, building a model to predict non-tippers is ethically problematic. It could lead to discriminatory practices where drivers avoid certain passengers, reducing accessibility to taxi services for those unable or unwilling to tip, potentially causing customer backlash and eroding trust.

Modified Objective: Due to the ethical concerns, the objective has been adjusted. Instead of predicting non-tippers, the goal is now to build a model that predicts whether a customer is likely to be a generous tipper (defined as tipping 20% or more of the fare).

Revised Purpose: The purpose of this modified model is to identify potentially high-revenue rides based on tipping propensity. This could potentially be used to inform strategies aimed at maximizing driver earnings in a way that is less ethically fraught than excluding potential passengers (though careful consideration of implementation is still required). The focus shifts from penalizing predicted non-tippers to potentially identifying rewarding opportunities.

- Who are your external stakeholders that I will be presenting for this project?

Primary: NYC Taxi & Limousine Commission (TLC) representatives (the client).

Secondary (Impacted Parties, not present in the Project Presentation):

Taxi Drivers: They are the intended users of the application derived from the model and are directly affected by its predictions and any resulting strategies. Their trust and acceptance are crucial.

Taxi Passengers/Customers: Their experience could be affected (positively or negatively depending on implementation) by how drivers use the model's insights. The original proposal posed a significant risk of negative impact (difficulty getting rides).

Taxi Fleet Owners/Operators: They have an interest in driver earnings, vehicle utilization, and overall service reputation.

- What resources do you find yourself using as you complete this stage?

Jupyter Notebook: The structured notebook provided for the project, including specific prompts and guidance.

PACE Strategy Document: This document itself, guiding the planning and reflection process.

Ethical Frameworks/Guidelines: Implicitly used to evaluate the initial request and justify the modification of the objective.

NYC TLC Data Dictionary/Metadata: Documentation describing the available data features.

Python Libraries (Conceptual): Anticipating use of Pandas, Scikit-learn, etc., during the Analyze, Construct, and Execute stages.

Prior Knowledge: Foundational concepts (EDA, statistics, visualization, Python).

- Do you have any ethical considerations at this stage?

Yes, significant considerations, primarily related to the initial request:

Discrimination/Bias: The original model predicting non-tippers risks reinforcing existing biases and leading to drivers discriminating against passengers based on predicted tipping behavior, potentially correlated with location, time, or other factors acting as proxies for demographics or socioeconomic status. This creates unequal access to essential transportation.

Consequences of Errors (Original Model):

- False Positive (Predicts "No Tip," but customer would have tipped): Driver avoids a fare they could have earned a tip on; customer is unjustly denied service. This harms both parties and the system's reliability.
- False Negative (Predicts "Tip," but customer doesn't): Driver may feel misled or frustrated, potentially leading to distrust in the app/system.
- Negative Impact on Drivers: Constant alerts about non-tippers could create a negative or stressful work environment.
- Negative Impact on Customers: Difficulty hailing cabs, particularly for those who cannot afford to tip, undermining the public service aspect of taxis.

- Trust and Reputation: Implementing such a feature could severely damage the reputation of the TLC and taxi services.

Ethical Considerations for the Modified Goal (Predicting Generous Tippers):

While less problematic than predicting non-tippers, care must still be taken. How will this information be used? If it leads to only prioritizing "generous" tippers and deprioritizing others, it could still lead to de facto discrimination. The implementation must be carefully designed to avoid negative consequences (e.g., using it for positive reinforcement rather than exclusion).

Transparency about how the model works and is used remains important

- Is my data reliable?

Source: NYC TLC data is generally considered the official record.

Tip data is unreliable, particularly for cash transactions. Tips paid in cash are often not recorded electronically and may appear as $0 in the dataset. This makes it extremely difficult to distinguish between an actual $0 tip and an unrecorded cash tip. This significantly impacts the ability to model tipping behavior accurately, especially for the original goal of predicting "no tip."

Also, missing values in other fields, potential inaccuracies in GPS/location data, meter data entry errors.

So, the data requires careful cleaning and preprocessing. The unreliability of the tip amount for cash transactions is a major limitation that must be acknowledged and addressed (e.g., potentially filtering data to only include credit card transactions where tips are electronically recorded, although this introduces its own biases).

- What data do I need/would like to see in a perfect world to answer this question?

Accurate, Verified Tip Amounts: For all transactions, regardless of payment method.

Customer Historical Data: Unique, anonymized customer IDs to track individual tipping history over time.

Trip Context: Real-time traffic, granular weather, specific events, driver interaction quality (subjective), vehicle condition.

Passenger Demographics (Anonymized): Tourist vs. local, general purpose of trip (business/leisure).

Driver Information: Experience level, average rating.

- What data do I have/can I get?

Standard NYC TLC Trip Record Data:

- Pickup/Dropoff Date & Time
- Pickup/Dropoff Location (Borough, Zone ID)
- Trip Distance

- Passenger Count
- Fare Amount, Tolls, Taxes, Surcharges
- Tip Amount (with the known reliability issue for cash)
- Total Amount
- Payment Type (Crucial for potentially filtering or segmenting the analysis)
- Rate Code ID
- Vendor ID

Engineered features:

- Mean Duration
- Mean Distance
- Predicted Fare
- Tip Percent
- Generous (Boolean)
- Day
- AM Rush
- Day time
- PM Rush
- Night time
- Month

- What metric should I use to evaluate success of my business/organizational objective? Why?

A little over half of the customers in this dataset were "generous" (tipped ≥ 20%). The dataset is very nearly balanced.

To determine a metric, I consider the cost of both kinds of model error:

False positives (the model predicts a tip ≥ 20%, but the customer does not give one)

False negatives (the model predicts a tip < 20%, but the customer gives more)

False positives are worse for cab drivers, because they would pick up a customer expecting a good tip and then not receive one, frustrating the driver.

False negatives are worse for customers, because a cab driver would likely pick up a different customer who was predicted to tip more—even when the original customer would have tipped generously.

The stakes are relatively even. We want to help taxi drivers make more money, but we don't want this to anger customers. Our metric should weigh both precision and recall equally.

F1 score is the metric that places equal weight on true positives and false positives, and so therefore on precision and recall.

PACE: Analyze Stage

- Revisit “What am I trying to solve?”Does it still work? Does the plan need revising?

Current Objective: We are trying to build a model to predict if a customer is likely to be a "generous tipper" (tipping >= 20% of the fare) using historical NYC TLC taxi trip data. This objective was modified from the original problematic request (predicting non-tippers) during the Plan stage.

Does it still work? Yes, this modified objective remains ethically preferable and potentially valuable for drivers/TLC.

Does the plan need revising? Yes, based on initial data exploration (EDA) during this Analyze stage, the plan needs significant revision regarding data filtering:

The Cash Tip Issue: EDA confirms that tip amounts for cash transactions are unreliable (often recorded as $0, impossible to distinguish from actual $0 tips). Calculating a meaningful tip percentage for these trips is impossible.

Required Revision: To create a valid target variable (tip >= 20%), we must filter the dataset to include only trips paid by credit card, where tips are electronically recorded. We also need to handle trips with $0 fares appropriately if they exist.

This narrows the scope of the model. It will predict generous tipping behavior only among credit card paying customers. This limitation and potential bias (credit card users might tip differently than cash users) must be clearly communicated. The goal is now more accurately: "Predict if a credit card paying customer will tip >= 20%."

- Does the data break the assumptions of the model? Is that ok, or unacceptable?

Intended Models: The project focuses on tree-based models (e.g., Random Forest, Gradient Boosting like XGBoost).

Tree Model Assumptions: These models are generally robust and have fewer strict assumptions compared to linear models:

Linearity: Not required.

Normality: Not required for features or residuals.

Homoscedasticity: Not required.

Potential Issues Identified in EDA & Their Acceptability:

Multicollinearity: EDA likely reveals high correlation between features like trip_distance and fare_amount. While high multicollinearity can slightly affect feature importance interpretability in trees, it doesn't typically break the model's predictive performance significantly. This is generally acceptable, but noted.

Data Leakage: EDA is crucial for identifying leakage. For instance, the total_amount feature directly includes the tip_amount. Using total_amount to predict a function of the tip would be leakage. This is unacceptable and leaky features must be removed.

Outliers: EDA will likely identify outliers (e.g., extremely long/short trips, unusually high fares). Tree models are relatively robust to outliers, but extreme values can sometimes influence splits. Depending on the nature/frequency, capping or investigating these outliers might be considered during feature engineering. Generally acceptable, but handle appropriately.

Missing Values: EDA quantifies missing data. Some tree algorithms (like XGBoost) can handle NaNs internally, while others require imputation. A strategy must be chosen based on the extent of missingness and the chosen algorithm. Acceptable if a clear handling strategy is implemented.

Feature Scaling: Not strictly required for standard tree algorithms, as they partition based on feature values. EDA confirms variable ranges, but scaling is usually omitted unless using specific regularized tree versions or combining with distance-based methods. Acceptable (lack of scaling is okay).

Conclusion: The data doesn't inherently break core tree-model assumptions in an unacceptable way, provided critical issues like data leakage are addressed by removing the problematic features. Other issues like multicollinearity, outliers, and missing values can be managed through standard preprocessing techniques identified during EDA.

- Why did you select the X variables you did?

Variables (features) were selected based on a combination of:

Domain Knowledge: Features intuitively expected to influence tipping behavior (e.g., cost of service, time/location context, passenger load).

Data Availability: Using columns present in the TLC dataset.

EDA Insights:

- Confirming feature distributions and identifying usable data types (numerical, categorical).
- Identifying features with sufficient variation and non-excessive missing values (after cleaning/filtering).
- Removing features identified as causing data leakage (e.g., total_amount).
- Feature Engineering: EDA informs the creation of new, potentially more predictive features derived from existing ones.

Examples of Selected Variables (Post-EDA & Filtering):

- mean_duration
- Mean_distance
- predicted_fare
- tip_percent
- day
- am_rush

- daytime
- pm_rush
- nighttime
- month

We also know from your EDA that customers who pay cash generally have a tip amount of $0. To meet the modeling objective, we sampled the data to select only the customers who pay with credit card.

- What are some purposes of EDA before constructing a model?

Grasp the structure, data types, volume, and granularity of the data.

Identify and quantify missing values, outliers, duplicates, inconsistencies, or erroneous data points.

Analyze the distribution of individual variables (histograms, density plots, summary statistics).

Explore relationships between variables (scatter plots, correlation matrices) and identify potential multicollinearity.

Understand the distribution and characteristics of the target variable (in this case, the binary 'generous tipper' flag), especially its **class balance**, which is crucial for model evaluation strategy.

Discover opportunities to create new, potentially more predictive features from existing ones (e.g., extracting time components, calculating speed).

Identify irrelevant, redundant, or leaky features that should be excluded from modeling.

Validate assumptions relevant to potential modeling techniques (though less critical for trees).

Identify data limitations (like the cash tip issue) that necessitate adjustments to the project scope or modeling approach (like filtering).

Look for patterns that might indicate potential biases in the data collection or representation.

- What has the EDA told you?

Confirmed the unreliability of cash tips, making it essential to filter the dataset to credit card payments only to create a meaningful target variable (tip >= 20%).

Allowed calculation of the binary target variable generous_tip (1 if tip >= 20%, 0 otherwise) for the filtered data.

Revealed the distribution of the target variable. A little over half of the customers in this dataset were "generous" (tipped ≥ 20%). The dataset is very nearly balanced.

Confirmed the possibility and potential value of creating features like mean_duration, daytime, month.

Confirmed some columns used for feature engineering must now be excluded, like the tip_amount and trip_distance.

Quantified missing data in specific columns, requiring an imputation or removal strategy.

- What resources do you find yourself using as you complete this stage?

- **Programming Language & Libraries:**
  - Python: The primary language.
  - Pandas: For data manipulation, cleaning, filtering, and descriptive statistics.
  - NumPy: For numerical operations, handling arrays.
  - Matplotlib & Seaborn: For creating visualizations (histograms, scatter plots, box plots, heatmaps) to explore data distributions and relationships.
- **Environment: Jupyter Notebook:** For interactive exploration, code execution, and documentation.
- **Documentation & Knowledge:**
  - Pandas, Matplotlib, Seaborn documentation: For function usage and examples.
  - Course 6 materials: Notes and concepts related to EDA.
  - Statistical knowledge: Concepts like mean, median, standard deviation, correlation, distribution types.
  - NYC TLC Data Dictionary: To understand the meaning of each column.
  - Prior project experience/examples: Referencing similar analyses.

PACE: Construct Stage

- Do I notice anything odd? Is it a problem? Can it be fixed? If so, how?

I noticed that if I include many parameters in my GridSeachCV to experiment with in order to find me the optimal ones, it takes a lot of computational time and power (up to 20-30 mins). To fix it, I minimize my parameter options and when my model is fit, I pickle it and have it available for next notebook kernel reruns.

- Which independent variables did you choose for the model, and why?

The independent variables (features) chosen for the model are all columns of my final dataframe except the target variable generous:

1. **VendorID** (Encoded)
2. **passenger_count** (int64)
3. **RatecodeID** (Encoded)
4. **PULocationID** (Encoded)
5. **DOLocationID** (Encoded)
6. **mean_duration** (float64)
7. **mean_distance** (float64)
8. **predicted_fare** (float64)
9. **day** (object)
10. **am_rush** (int64 - binary flag)
11. **daytime** (int64 - binary flag)
12. **pm_rush** (int64 - binary flag)
13. **nighttime** (int64 - binary flag)
14. **month** (object)

Why these variables were chosen:

These variables were selected because they capture different aspects of the taxi trip that could plausibly influence a customer's tipping behavior (specifically, whether they tip generously >= 20%):

- Trip Characteristics (passenger_count, RatecodeID, mean_duration, mean_distance, predicted_fare):

passenger_count: More passengers might correlate with different trip types or perceived service levels.

RatecodeID: Different rate codes signify different trip types (standard, airport, negotiated) which often have different tipping norms.

mean_duration, mean_distance, predicted_fare: These engineered features represent the core service being paid for – its length, duration, and estimated cost. The fare amount itself is a primary factor against which the tip percentage is calculated, and longer/more expensive trips might have different tipping patterns.

- Location Context ((PULocationID, DOLocationID):

Pickup and dropoff locations are very important. Tipping habits can vary significantly based on neighborhood characteristics (e.g., business districts, residential areas, tourist hubs, airports, different boroughs).

- Temporal Context (day, month, am_rush, daytime, pm_rush, nighttime):

day, month: Day of the week and month capture weekly patterns (weekends vs weekdays) and seasonality (holidays, weather influences) which can affect travel purpose and tipping.

am_rush, daytime, pm_rush, nighttime: These engineered time-of-day features capture different contexts – commuting hours, midday trips, evening outings – which might correlate with different passenger types or trip purposes, influencing tipping.

- Operational Context (Vendor ID)

Represents the taxi technology provider. Different systems might have slightly different interfaces or processes, potentially subtly influencing the payment experience.

In essence, these features provide a multi-faceted view of the trip context (what, when, where, how much) without including variables that would directly leak information about the tip itself (like total_amount or the raw tip_amount).

- How well does your model fit the data? What is my model's validation score?

The model demonstrates a moderate fit to the data. The consistency between the cross-validation (CV) scores and the test set scores is good – they are very close (e.g., F1: 0.714 CV vs 0.723 test; Accuracy: 0.680 CV vs 0.687 test). This indicates that the model generalizes well to unseen data and isn't significantly overfit to the training data used during cross-validation.

The model's average validation scores from cross-validation (the RF CV row) are:

- Precision: ~0.675
- Recall: ~0.757
- F1-Score: ~0.714
- Accuracy: ~0.680

- Can you improve it? Is there anything you would change about the model?

Yes, likely. An F1-score of ~0.71-0.72 is decent, but often there's room for improvement in machine learning tasks.

Potential areas for improvement or changes include:

Hyperparameter Tuning: Perform more extensive tuning of the Random Forest's hyperparameters (e.g., n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features, potentially class_weight if imbalance is an issue). The current results might be from default or lightly tuned parameters.

Try Different Algorithms: Compare the Random Forest performance to other models, especially gradient boosting machines like XGBoost or LightGBM, which often achieve higher performance.

Feature Engineering: Revisit feature engineering. Could different features or interactions be created? Were the categorical encodings optimal?

- What resources do you find yourself using as you complete this stage?

Python Libraries:

- Scikit-learn: For RandomForestClassifier, train_test_split, cross_val_score (or similar CV functions), evaluation metrics (precision_score, recall_score, f1_score, accuracy_score, classification_report, confusion_matrix), and potentially hyperparameter tuning tools (GridSearchCV, RandomizedSearchCV).
- Pandas & NumPy: For final data manipulation before feeding into the model.

Conceptual Knowledge:

Understanding of the Random Forest algorithm, cross-validation principles, evaluation metrics, and the difference between validation and test sets.

Development Environment:

Jupyter Notebook or an IDE to write and execute the code.

Documentation:

Scikit-learn's official documentation for specific function parameters and usage examples.

PACE: Execute Stage

- What key insights emerged from your model(s)? Can you explain my model?

The most important factors for predicting a generous tip (>=20%) in my Random Forest model are VendorID, the predicted_fare, mean_duration, and mean_distance.

Random Forest model is an ensemble of many decision trees. It looks at combinations of the input features (like vendor, fare, duration, location, time) and learns rules from the historical data to decide whether a trip is likely to result in a tip of 20% or more. The feature importance plot shows which factors the model relied on most heavily to make these decisions.

- What are the criteria for model selection?

I compared the performance of Random Forest (RF) and XGBoost (XGB) using cross-validation and test set scores.

The primary criterion appears to be the F1-score on the test set. I noted the RF model had a slightly higher F1 score (~0.723) compared to the XGBoost model (~0.710) on the test data, leading me to choose Random Forest as the "champion".

- Does my model make sense? Are my final results acceptable?

Yes, the model makes intuitive sense. It's logical that factors related to the trip's cost and length (predicted_fare, mean_distance, mean_duration) would influence tipping percentage. The high importance of VendorID is interesting and suggests potential differences in user interface or processes between vendors.

Acceptable Results: Based on my analysis, I deemed the results "acceptable" and "satisfactory". An F1 score of ~0.72 means the model achieves a reasonable balance between precision and recall. While not perfect, predicting human behavior like tipping is inherently difficult. The performance is moderate.

- Do you think your model could be improved? Why or why not? How?

Yes, potentially. While acceptable, an F1 score of 0.72 suggests room for improvement.

How:

- Hyperparameter Tuning: Further tuning of the Random Forest parameters might yield better results.
- Feature Engineering: Explore creating more complex features or interactions (e.g., interactions between time and location).
- Different Models: Explore different variations of Ensemble models like adjusted versions of Random Forest.

- Were there any features that were not important at all? What if you take them out?

Least Important: Features like month_oct, day_sunday, day_saturday, day_tuesday, pm_rush appear at the very bottom of your importance plot with low values.

Removing these least important features might:

- Simplify the model slightly, making it faster to train and potentially easier to interpret.
- Have minimal impact (positive or negative) on performance if they truly contribute little information.
- Slightly decrease performance if they contribute minor but useful interaction effects.

It would require retraining and re-evaluating the model without these features to know the exact effect.

- What business/organizational recommendations do you propose based on the models built?

Focus on Service Quality: Since fare, duration, and distance are important, reinforce the idea that providing good service, especially on longer/more expensive trips (where tips constitute a larger absolute amount), is key.

Investigate VendorID Impact: The high importance of VendorID warrants investigation. Does one vendor's app/payment system encourage tipping more effectively (e.g., better UI, clearer prompts, higher default suggestions)? The TLC could explore standardizing best practices.

Context Matters: Acknowledge that location and time of day influence tipping, but don't use this to screen passengers (sticking to the ethical reframing). Insights could potentially inform drivers about generally higher-tipping areas/times if presented carefully and ethically.

Avoid Misuse: Strongly advise against using the model to "alert" drivers to potentially non-generous tippers, as this risks reintroducing the ethical issues of the original request (discrimination, reduced access). Focus on understanding drivers of positive outcomes.

- Given what you know about the data and the models you were using, what other questions could you address for the team?

- Can we predict the actual tip percentage or tip amount (regression task) instead of just a binary "generous" flag?
- Does the importance of features change significantly depending on the Pickup Location (e.g., are time features more important for airport pickups vs. Manhattan pickups)?
- How do different RatecodeID types influence tipping generosity?
- Can we build separate models for different segments (e.g., weekday vs. weekend, different boroughs) to get more nuanced insights?

- What resources do you find yourself using as you complete this stage?

- Python Libraries: scikit-learn (for feature_importances_, classification_report, confusion_matrix, ConfusionMatrixDisplay), matplotlib/seaborn (for plotting), pandas (for results tables).
- Model Object: The trained rf1.best_estimator_ object itself.
- Knowledge Base: Understanding of evaluation metrics (Precision, Recall, F1, Accuracy), confusion matrices, feature importance concepts, and the specific business context (NYC Taxis, tipping).
- Jupyter Notebook: To run the code and display results.

- Is my model ethical?

The model predicting generous tippers is more ethical than the original request to predict non-tippers.

However, ethics depend heavily on implementation. If the model is used purely for insights (like understanding vendor differences) or potentially for positive reinforcement in an equitable way, it can be ethical.

If it's used to prioritize service towards predicted generous tippers, potentially leading to neglect or avoidance of others, it could become unethical by creating unequal access. Clear guidelines for use are essential.

- When my model makes a mistake, what is happening? How does that translate to my use case?

Type I Error (False Positive): The model predicts a "generous tip" (Predicted=1), but the actual tip was not generous (True=0). (Count = 602 in my matrix).

Use Case Translation: The system might indicate a passenger is likely to be generous, but the driver receives a standard or low tip. This could lead to mild disappointment for the driver if they were expecting more based on the prediction. As you noted, this is more common.

Type II Error (False Negative): The model predicts not a generous tip (Predicted=0), but the actual tip was generous (True=1). (Count = 355 in my matrix).

Use Case Translation: The system gives no indication of a generous tip, but the driver receives one anyway. This results in a pleasant surprise for the driver.