POINT OF INTEREST DEDUPLICATION

TELENAV

13

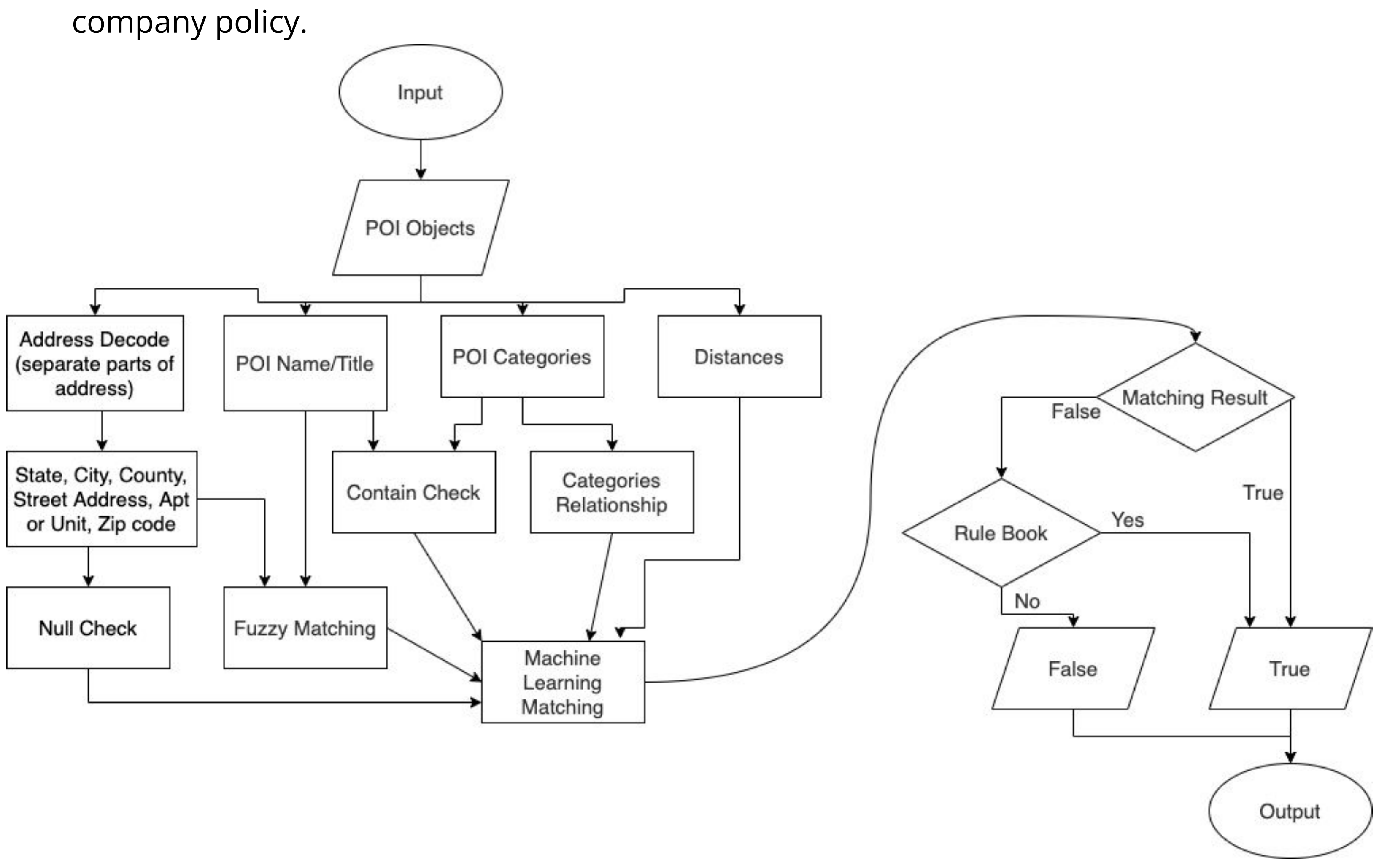
STUDENTS: AADITYA DESAI, ADIT JHA, BING SYUAN WANG, KHOA TRAN

Background and Objectives

- A point of interest is a specific physical location, which someone may find interesting. Examples: Restaurants, retail and grocery stores, gas stations etc.
- Telenav provides GPS satellite navigation, local searches, entertainment for automotive navigation.
- Points of interest are provided to Telenav by different vendors. As a result, many provided points of interest are different in title, address, etc., but are duplicates, and vice versa
- Examples: POI 1: "Costco", POI 2: "Costco Wholesale" ⇨ TRUE (duplicates)
  POI 1: "Costco", POI 2: "Costco Gas" ⇨ FALSE (non-duplicates)
- Objective:
  - Develop and integrate into Telenav data processing pipeline a solution to find and remove POI duplications from Telenav datasets
  - Machine Learning model prediction accuracy above 95%
  - Java API library with a function entrypoint and Command Line Interface for analyzing large POI datasets

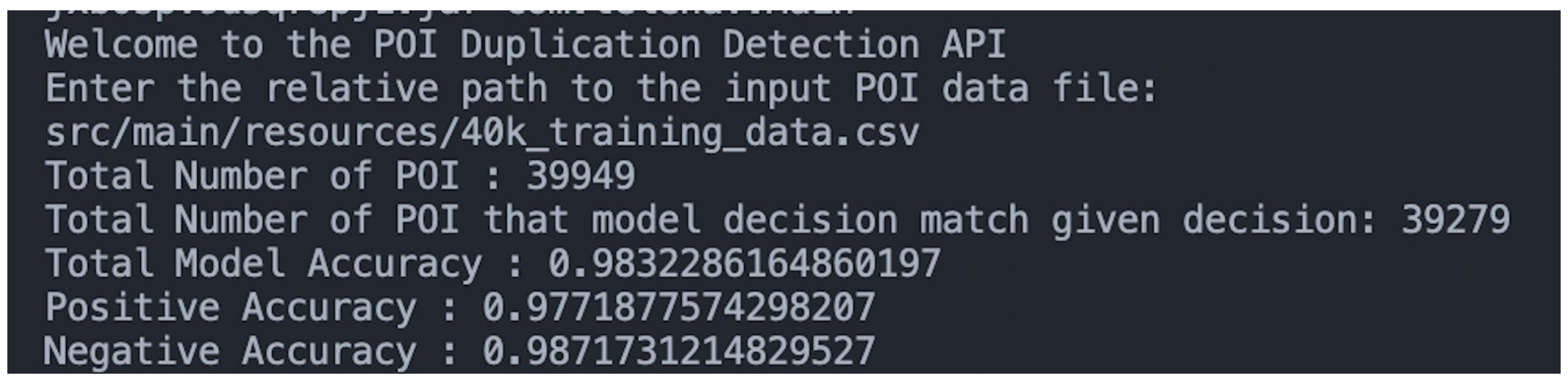
Pre-processing and Features

- The overall workflow involves three large steps. The first is data pre-processing to extract features such as title/address similarity, physical distance, category relationships, etc.
- The second is the prediction made by the machine learning model using the extracted features. The third is the hard-coded rulebook that overrules model decisions based on company policy.



Command Line Interface

- One of our supporting deliverables is to implement a command line interface for Telenav to perform offline testing
- CLI request a CSV input file with POI pairs and the given decision of being duplicates
- For each POI pair, CLI triggers a call to the Java API function to determine if the POIs are duplicates
- Outputs the accuracy of the model based on number of decisions made by the model that matches the given decision
- Outputs a text file with inference results for each API call named output.csv



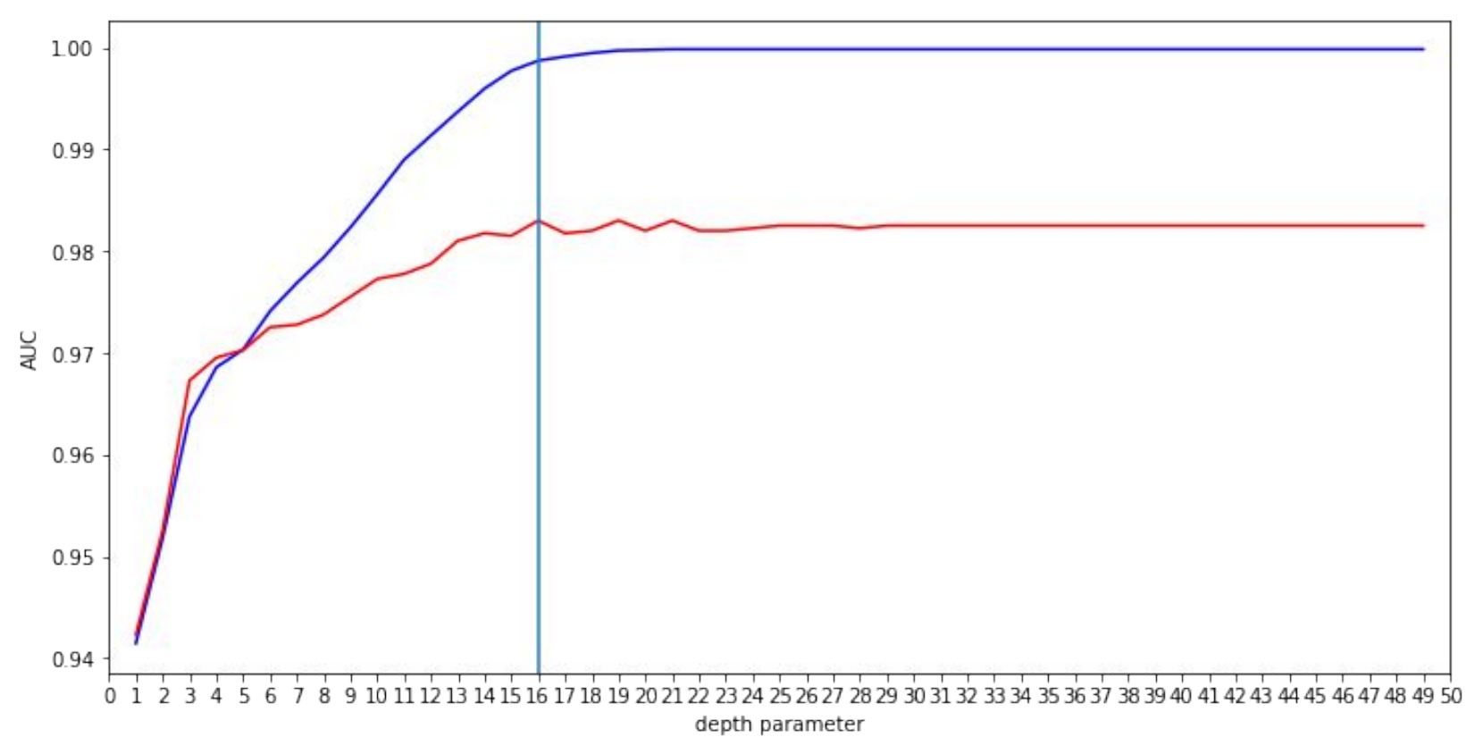
```
Welcome to the POI Duplication Detection API
Enter the relative path to the input POI data file:
src/main/resources/40k_training_data.csv
Total Number of POI : 39949
Total Number of POI that model decision match given decision: 39279
Total Model Accuracy : 0.9832286164860197
Positive Accuracy : 0.9771877574298207
Negative Accuracy : 0.9871731214829527
```

Results

- Overall, we were able to meet and in fact exceed accuracy expectations.
- The final accuracy scores with all of the features and rulebook for the API library are:
  - Overall Accuracy = **98.3%**
  - Positive Accuracy = **97.7%**
  - Negative Accuracy = **98.7%**

Machine Learning Research

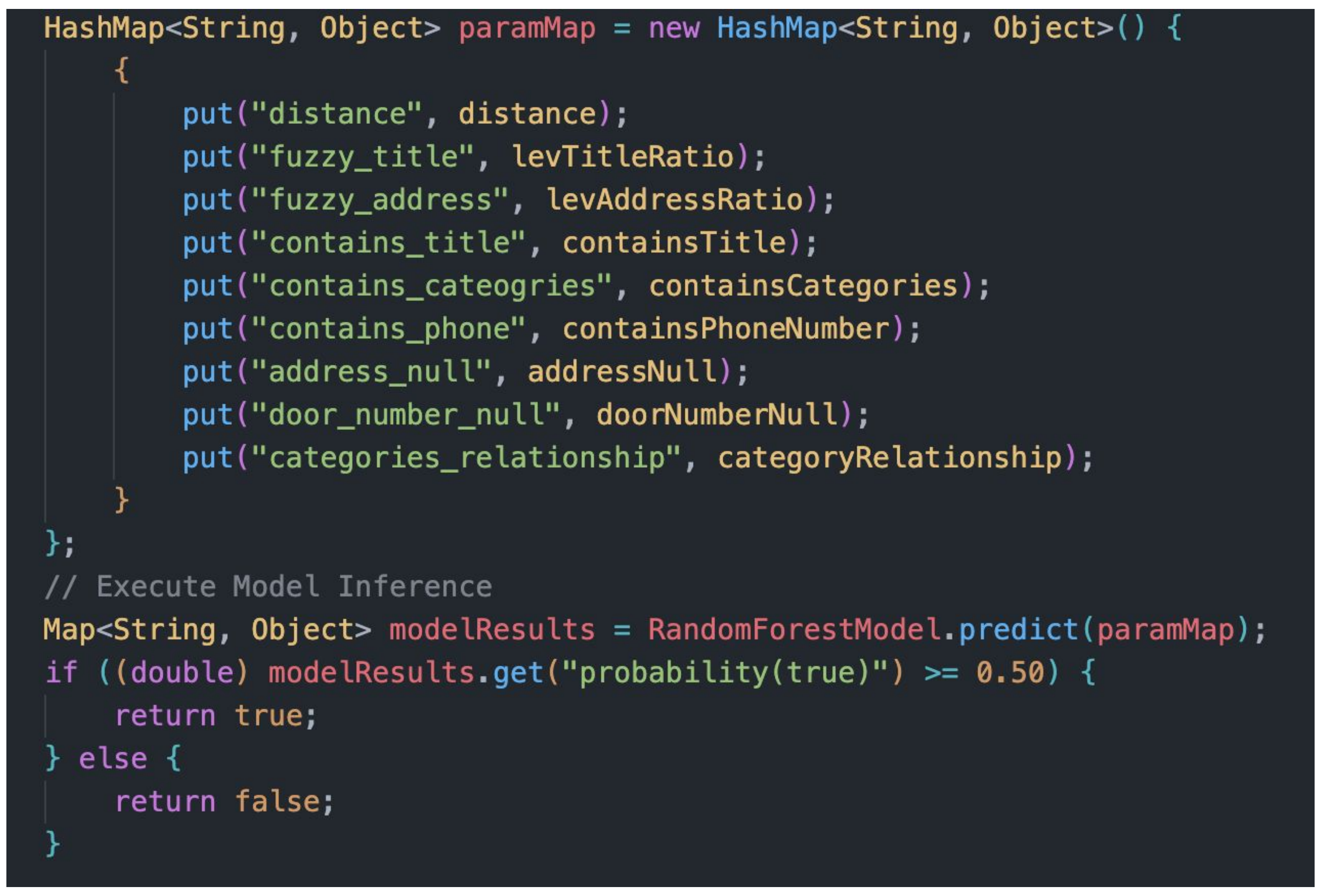
- Since our training target is categories and the training data has been labeled, we use the classification model to train the data.
- After trying many different models, we finally settled on the Random Forest model and used ROC, AUC to validate and tune hyperparameters and accuracy.
- Once the Python model is trained, we convert it to a Java model to be used.



	Accuracy	Total Accuracy		Positive Accuracy		Negative Accuracy	
		Training	Validation	Training	Validation	Training	Validation
Model	Logistic Regression	95.7%	95.6%	83.3%	82.1%	97.6%	97.6%
	Random Forest	**97.4%**	**96.1%**	**88.6%**	**82.4%**	**98.8%**	**98.1%**
	Decision Tree	96.1%	95.8%	84.4%	82.1%	97.9%	97.8%
	Extra Tree	96.8%	96.0%	86.5%	81.5%	98.4%	98.0%
	Bagging	99.2%	95.8%	96.5%	79.7%	99.6%	98.1%
	Bagging + KNeighbors	96.9%	95.5%	86.3%	79.8%	98.5%	97.8%
	Bagging + SVC	95.6%	95.5%	78.7%	77.7%	98.1%	98.1%
	SGD	95.8%	95.7%	81.8%	80.1%	97.9%	97.9%

Java Library Implementation

- The Java API Library is the main deliverable for Telenav as it will serve as the basis for executing deduplication analysis at scale.
- The grouping function kicks off by first analyzing whether the POI objects contain any category pairs that are considered non-pairs by the built rulebook.
- If the rulebook passes, the function performs data normalization to prepare the POI attributes for feature analysis.
- A series of helper functions are called which calculate the feature parameter values used for the Random Forest model.
- The library will then return true or false for the POI comparison if the Random Forest model has a true confidence score of at least 50%.



```
HashMap<String, Object> paramMap = new HashMap<String, Object>() {
    {
        put("distance", distance);
        put("fuzzy_title", levTitleRatio);
        put("fuzzy_address", levAddressRatio);
        put("contains_title", containsTitle);
        put("contains_cateogries", containsCategories);
        put("contains_phone", containsPhoneNumber);
        put("address_null", addressNull);
        put("door_number_null", doorNumberNull);
        put("categories_relationship", categoryRelationship);
    }
};
// Execute Model Inference
Map<String, Object> modelResults = RandomForestModel.predict(paramMap);
if ((double) modelResults.get("probability(true)") >= 0.50) {
    return true;
} else {
    return false;
}
```

Future Development Ideas

- Provided Telenav with Python Jupyter Notebook to further train the model based on new data, parameters, etc.
- Add new preprocessing steps to provide more insight to model for inference
- Analyze the text similarity scoring system for non-english native POI titles for improvement on difficult edge cases
- Utilize more information from the POI attributes to either rule out or improve positive accuracy

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY *of* WASHINGTON

FACULTY MENTOR: Karthik Mohan

INDUSTRY MENTOR: Akira Zhang

SPONSORS: TELENAV
ELECTRICAL & COMPUTER ENGINEERING DEPARTMENT, UNIVERSITY OF WASHINGTON