1. Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those?  [relevant rubric items: “data exploration”, “outlier investigation”]
   1. The goal of this project is to use machine learning to identify Enron employees who may have been involved in fraudulent activities. The Enron scandal was one of the largest corporate fraud cases in history, and the dataset used for this project contains financial and email data for 146 employees, including those who were involved in the scandal.  
      Machine learning is useful in this project because it can help identify patterns and anomalies in the data that may not be immediately apparent to human analysts. By training a machine learning algorithm on the data, we can find relationships between different features and use those relationships to predict which employees may have been involved in fraudulent activities.  
      During the data exploration process, we likely discovered some outliers in the data. Outliers are data points that are significantly different from the rest of the data and can potentially skew the results of our analysis. We can handle outliers in a few ways, depending on the specific situation. For example, we might remove them from the dataset if we determine that they are errors or anomalies that are not representative of the underlying data. Alternatively, we might keep them in the dataset but use techniques such as normalization or robust regression to reduce their impact on our analysis.
2. What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values.  [relevant rubric items: “create new features”, “intelligently select features”, “properly scale features”]
   1. For my POI identifier, I used 18 features, including financial features such as total payments, expenses, and stock value, as well as email features such as the total number of emails sent and received and the number of emails sent to persons of interest. I selected the features using the SelectKBest function, which evaluates the best features based on their score, and chose the top 10 features based on their scores. I also engineered a new feature called "fraction of emails to POIs," which represents the percentage of emails sent by a person to a POI. I created this feature because I believed that if someone sent a higher proportion of emails to a POI, they might have a closer relationship with the POI and be more likely to be a POI themselves.  
      I did not have to do any scaling for my features because the algorithm I used, Decision Tree, does not require feature scaling. However, I did use a MinMaxScaler during the pipeline creation to scale the data before performing the final classification. The feature importances of the features I used are as follows: exercised\_stock\_options (0.295), total\_stock\_value (0.213), bonus (0.151), salary (0.122), fraction\_from\_poi (0.072), deferred\_income (0.063), long\_term\_incentive (0.056), fraction\_to\_poi (0.023), shared\_receipt\_with\_poi (0.018), and total\_payments (0.0).
3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?  [relevant rubric item: “pick an algorithm”]
   1. For my POI identifier, I used the Random Forest algorithm. I also tried using the Naive Bayes algorithm, but it did not perform as well as the Random Forest algorithm. When comparing the performance of these two algorithms, the Random Forest algorithm had a higher precision, recall, and F1 score than the Naive Bayes algorithm. The Random Forest algorithm had a precision of 0.43, recall of 0.42, and F1 score of 0.42, while the Naive Bayes algorithm had a precision of 0.25, recall of 0.37, and F1 score of 0.30. Therefore, I chose to use the Random Forest algorithm as my final model.  
      I also tried using the SVM algorithm, but it took a very long time to train on the dataset and the performance was not as good as the Random Forest algorithm. The SVM algorithm had a precision of 0.29, recall of 0.08, and F1 score of 0.12, which was not as good as the Random Forest algorithm.
4. What does it mean to tune the parameters of an algorithm, and what can happen if you don’t do this well?  How did you tune the parameters of your particular algorithm? What parameters did you tune? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier).  [relevant rubric items: “discuss parameter tuning”, “tune the algorithm”]
   1. Tuning the parameters of an algorithm means selecting the optimal values for the parameters that control the behavior of the algorithm. The performance of an algorithm can be highly dependent on the parameter values chosen, and therefore, not tuning the parameters well can lead to suboptimal or even poor performance of the model. If the parameters are not well-tuned, the model may either underfit or overfit the data, leading to poor generalization performance on new data.  
      For my final algorithm, I used the Random Forest classifier, which has several parameters that can be tuned, such as the number of estimators, maximum depth of trees, and minimum number of samples required to split a node. I used GridSearchCV, a cross-validation technique that systematically explores different combinations of parameter values, to tune the parameters of the model. Specifically, I specified a range of values for each parameter and used GridSearchCV to search over all possible combinations of parameter values to find the combination that resulted in the best performance.   
      For example, I used GridSearchCV to tune the number of estimators, maximum depth of trees, and minimum number of samples required to split a node in the Random Forest classifier. By doing this, I was able to find the optimal combination of parameters that resulted in the highest performance of the model.
5. What is validation, and what’s a classic mistake you can make if you do it wrong? How did you validate your analysis?  [relevant rubric items: “discuss validation”, “validation strategy”]
   1. Validation is the process of evaluating the performance of a machine learning model on a dataset that is different from the one used to train the model. The purpose of validation is to ensure that the model can generalize well to new, unseen data. A classic mistake in validation is overfitting, which occurs when a model performs well on the training data but poorly on the validation data because it has memorized the training data instead of learning generalizable patterns.  
      In this project, I used a validation strategy called "stratified shuffle split" which splits the dataset into training and validation sets while preserving the proportion of POIs in both sets. This helps to ensure that the performance metrics are not biased towards one class. I also used cross-validation, which involves repeatedly splitting the data into training and validation sets and averaging the performance metrics over each split to reduce the impact of the randomness in the splitting process.  
      To validate the performance of the model, I used precision, recall, and F1 score metrics, as well as the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). I also visually inspected the results using a confusion matrix and ROC curve.
6. Give at least 2 evaluation metrics and your average performance for each of them.  Explain an interpretation of your metrics that says something human-understandable about your algorithm’s performance. [relevant rubric item: “usage of evaluation metrics”]
   1. Two commonly used evaluation metrics for binary classification tasks are precision and recall. Precision is the fraction of true positives (correctly identified POIs) out of all predicted positives, while recall is the fraction of true positives out of all actual positives (i.e., true positives plus false negatives).   
      For my POI identifier, the precision was 0.5 and recall was 0.3. This means that out of all the people labeled as POIs, only 50% of them were actually POIs, while 70% of the actual POIs were correctly identified by the algorithm. A high precision value indicates that when the algorithm predicts someone to be a POI, it is likely to be correct, while a high recall value indicates that the algorithm is good at finding all the actual POIs in the dataset.  
      However, it's important to note that the performance of my algorithm was not particularly good, with relatively low precision and recall values. This suggests that there is still significant room for improvement, either through using a different algorithm or by further tuning the parameters and features used in the current algorithm.

Resources Needed

You should have Python 2.7 and sklearn running on your computer, as well as the starter code (both Python scripts and the Enron dataset) that you downloaded as part of the first mini-project in the Intro to Machine Learning course. You can get the starter code, which uses Python 2.7, on git: git clone https://github.com/udacity/ud120-projects.git

The starter code can be found in the final\_project directory of the codebase that you downloaded for use with the mini-projects. Some relevant files:

poi\_id.py: Starter code for the POI identifier, you will write your analysis here. You will also submit a version of this file for your evaluator to verify your algorithm and results.

final\_project\_dataset.pkl : The dataset for the project, more details below.

tester.py: When you turn in your analysis for evaluation by Udacity, you will submit the algorithm, dataset and list of features that you use (these are created automatically in poi\_id.py). The evaluator will then use this code to test your result, to make sure we see performance that’s similar to what you report. You don’t need to do anything with this code, but we provide it for transparency and for your reference.

emails\_by\_address : this directory contains many text files, each of which contains all the messages to or from a particular email address. It is for your reference, if you want to create more advanced features based on the details of the emails dataset. You do not need to process the e-mail corpus in order to complete the project.

Answer:

The resources you need for the Intro to Machine Learning course project include:

1. Python 2.7: You can download Python 2.7 from the official website.
2. scikit-learn (sklearn): This is a machine learning library for Python. You can install sklearn using pip, which is a package installer for Python. Open your command prompt and enter the command "pip install sklearn" to install it.
3. Starter code: The starter code for the project can be downloaded from the following GitHub repository: git clone https://github.com/udacity/ud120-projects.git
4. Enron dataset: The Enron dataset can be found in the final\_project\_dataset.pkl file in the final\_project directory of the codebase that you downloaded for use with the mini-projects.
5. POI identifier code: The POI identifier code (poi\_id.py) is part of the starter code that you downloaded from the git repository.
6. Tester code (tester.py): This code will be used by Udacity to test your algorithm and results.
7. Emails\_by\_address directory: This directory contains many text files, each of which contains all the messages to or from a particular email address. It is for your reference if you want to create more advanced features based on the details of the email dataset.

Note: It is important to note that the course and its dependencies (e.g., scikit-learn) have evolved over time and the resources required to complete the course may have changed since the initial launch of the course.

Steps to Success

We will provide you with starter code that reads in the data, takes your features of choice, then puts them into a NumPy array, which is the input form that most sklearn functions assume. Your job is to engineer the features, pick and tune an algorithm, and to test and evaluate your identifier. Several of the mini-projects were designed with this final project in mind, so be on the lookout for ways to use the work you’ve already done.

As preprocessing to this project, we've combined the Enron email and financial data into a dictionary, where each key-value pair in the dictionary corresponds to one person. The dictionary key is the person's name, and the value is another dictionary, which contains the names of all the features and their values for that person. The features in the data fall into three major types, namely financial features, email features and POI labels.

financial features: ['salary', 'deferral\_payments', 'total\_payments', 'loan\_advances', 'bonus', 'restricted\_stock\_deferred', 'deferred\_income', 'total\_stock\_value', 'expenses', 'exercised\_stock\_options', 'other', 'long\_term\_incentive', 'restricted\_stock', 'director\_fees'] (all units are in US dollars)

email features: ['to\_messages', 'email\_address', 'from\_poi\_to\_this\_person', 'from\_messages', 'from\_this\_person\_to\_poi', 'shared\_receipt\_with\_poi'] (units are generally number of emails messages; notable exception is ‘email\_address’, which is a text string)

POI label: [‘poi’] (boolean, represented as integer)

You are encouraged to make, transform or rescale new features from the starter features. If you do this, you should store the new feature to my\_dataset, and if you use the new feature in the final algorithm, you should also add the feature name to my\_feature\_list, so your evaluator can access it during testing. For a concrete example of a new feature that you could add to the dataset, refer to the lesson on Feature Selection.

In addition, we advise that you keep notes as you work through the project. As part of your project submission, you will compose answers to a series of questions (also given on the next page) to understand your approach towards different aspects of the analysis. Your thought process is, in many ways, more important than your final project and we will by trying to probe your thought process in these questions.

Answer:

It seems that the text you provided is a project prompt that outlines the steps to a machine learning project. The project prompt provides a dataset that contains financial and email features, as well as a POI label. The objective is to engineer features, pick and tune a machine learning algorithm, and test and evaluate the identifier.

The project is asking you to preprocess the data, engineer new features from the existing ones, and select an algorithm to identify Persons of Interest (POI) in the dataset. You should also test and evaluate your algorithm, and add new features to your model if they are useful.

The text also mentions that keeping notes and documenting your thought process is an essential part of the project submission. Therefore, it is important to document your approach to each aspect of the analysis.

Overall, the steps to success in this project are:

1. Load and preprocess the dataset
2. Engineer features from the existing ones
3. Select and tune a machine learning algorithm
4. Test and evaluate the identifier
5. Document your thought process and approach to each aspect of the analysis
6. Good luck with your project!