Aspiring Minds

Helping university career services target the right mix of companies

A staggering 10 million students were enrolled in engineering programs across India in 2016. With no concept of career fairs in Indian engineering colleges, campus recruitment drives (known as placements) are an integral part of the Indian education system. A separate department, commonly referred to as the Placement Cell, is responsible for inviting companies every year during what is called the 'Placement Season' to hire students who are about to graduate. However, Placement Cells, in general, fail to respond to the ever-changing diversity in student profiles and career interests, and stick to a static list of companies for the Placement Season every year. With this project, we created a machine learning model that would help the Placement Cell to be better prepared for an incoming Placement Season. By using student academic scores, graduation and job information, we populate the list of job domains and probable student distribution that the Placement Cell could use and target companies in order to cater to the requirements of the current student profiles. Additionally, we use existing salary information to predict salary ranges for a given student profile; information that the Placement Cell can use to negotiate salary packages with the invited companies and job profiles. Our results report reasonably high accuracies in predicting job domains and salary buckets. We believe our model can help Placement Cells target the right mix of companies thus enabling students to find better jobs.

Abhishek Sinha, Arnav Goyal, Rohit Raghavan, Sandeep Pal

INFO 251: APPLIED MACHINE LEARNING

Introduction

Motivation

The Indian education system is quite different from the education system in the US, especially when it comes to the job search process of graduating students. In the US, universities and colleges organize career fairs, alumni meetups and company talks for students to find job opportunities through networking. In contrast, Indian universities and colleges invite various companies on campus to hire students. The invited companies commonly conduct technical & aptitude tests, group discussions and face-to-face interviews before hiring students. This process of inviting companies on campus to hire students is popularly known as 'campus placement'. As undergraduate students from India, we have been through the campus placement process where we applied to several companies that were invited to our engineering institutions.

The campus placement process in every Indian institution starts with a 'Placement Cell'. A placement cell comprises of college officials and students, and is responsible for contacting companies and invite them to conduct their recruitment process to hire students who are in their final year. Generally, institutes have long-standing relationships with a few companies which visit that institution every year. This happens because an institution develops a good reputation of having students that do well in those companies. In spite of this, the number of students that a company hires from a particular college can greatly vary from year to year.

After talking to a couple of placement cell officers, we found that that colleges don't have a specific approach to inviting companies and they tend to invite the same companies every year without thinking too much about inviting new companies. Thus, they fail to cater to the ever-changing student profiles and not every targeted domain is covered in the process. We felt that machine learning could help placement cells in taking more informed decisions about the type of companies that should be invited. This informed decision would take into consideration the scores and qualities of the students.

With this goal in mind, our search for a dataset with student academic scores and job outcomes led us to the Aspiring Minds Dataset. The dataset was released by Aspiring Minds, a test taking institution similar to the GRE or GMAT, but who administer tests of job skills. These standardized score are supposedly a good indicator of a student's performance on the job, and are considered by companies during their recruitment process. We felt that this dataset would be perfect for our task of predicting possible job domains that students might find jobs in, given their academic scores. Armed with a dataset of student's academic scores, the placement cell of an institution could use this model to estimate the number and job domains their students would likely find jobs in. This could be the basis for a more targeted strategy by placement cells for reaching out to companies that offer such jobs, thus maximizing campus placements of their students.

Prior Work

The AMCAT dataset was part of a data challenge organized by ACM IKDD Conference on Data Science in 2016. The primary task of the data challenge was predicting salaries. Subtasks consisted of drawing insights on what factors determined salaries and creating interesting visualizations on job titles and job cities.

The papers submitted by the four winning teams are linked in the Appendix. Following is a short summary of the methods used in each paper:

1. After dividing the salary on 5 bins, the authors used different regression models - ridge regression, lasso regression and linear regression with median of means estimator to predict salaries. They used model coefficient values to infer factors affecting salary.
2. The authors use three different models - multiple linear regression, SVM and multivariate adaptive regression splines to make salary predictions. The also built a Salary Predictor app from their model.
3. The approach taken in predicting salaries consisted of dividing salaries into three buckets: < ₹300,000, ₹300,001 - ₹400,000 and >₹400,001 and applying various supervised machine learning models. The authors used Random Forests node analysis and feature importances to infer factors affecting salary. They also used Pearson Correlation test and Welch Two Sample t-test to find and make inferences from features correlated to salary.
4. The authors used bayesian networks to predict salary and draw inferences.

To our knowledge, no prior work has been done on predicting job domains and corresponding salary buckets, the task we undertake in this project.

Dataset Description and Preprocessing

Description

The dataset contains information about engineering students and their employment outcomes. Each record of the dataset consists of an anonymized students profile/academic scoring information along with their employment outcome information. There are a total of 3998 records, and 39 features.

Candidate Profile Information includes:

- Scores on Aspiring Minds' AMCAT – a standardized test of job skills. The test includes cognitive, domain and personality assessments
- Personal information like gender, date of birth, etc.
- Pre-university information like high school grades, high school location
- University information like GPA, college major, college reputation proxy.
- Demographic information like location of college, candidates' permanent location

Employment Outcome Information includes:

- First job annual salary
- First job title
- First job location

Preprocessing

Given that our dataset consisted of students' self reported information in freeform text fields, a major chunk of our time was spent in sanitizing the dataset. We cleaned our data manually as follows:

1. Categorical text features
   a. **Specialization** - There were 46 unique entries, may of which were the similar (eg. '*computer engineering*' vs. '*computer science & engineering*' or '*electronics & telecommunication*' vs. '*electronics and communication engineering*'). We mapped these into 10 unique entries.

   b. **10board and 12board** - Indicates the board the student's school follows. India has two central boards and every state with its own board. We mapped 340 unique values into 3 values: 2 central boards ('*cbse*', '*icse*') and '*state board*'.

   c. **JobCity** - After fixing entries with typos (of which there were many), we collated JobCity by area eg. a single area '*NCR*' covering *'Delhi', 'Gurgaon', 'Noida', 'Ghaziabad'*. (This is similar to the Bay Area representing SF, East Bay, Berkeley etc.). We dropped 10 records where the JobCity was located outside India.

   d. **Domain** - The free-form input for the designation field made it very difficult to use the original values. We combined a list of ~420 designations into 11 job domains. For instance, designations such as Software Engineering, Full Stack Developer were grouped into the 'Comp. Science Engineering' domain.

We then one-hot encoded the categorical features.

2. Numeric features
    a. ComputerProgramming, ElectronicsAndSemicon, ComputerScience, MechanicalEngg, ElectricalEngg, TelecomEngg had very few values since they are optional components of the AMCAT exam. We created a new feature with the the average scores of the above features and mapped values 0 (did not take exam) and 1 (took exam) in the original feature columns.
3. Date features
    a. DOB was converted to Age
    b. DOJ (Date of joining) and DOL (Date of leaving) was converted to a new field FirstJobDuration.
    c. DOJ and Year of Graduation was used to create a new feature denoting the number of years of experience the candidate has

Methods

Exploratory Data Analysis

We conducted an initial exploratory data analysis to understand the distribution of data and identify trends that could be useful in feature engineering for our machine learning models. Some of the graphs we plotted during EDA are listed in Appendix B. We concluded the following:

Looking at the distribution of students in our custom engineered feature domains (Fig B1a and B1b), we realized that data is imbalanced as a large percentage of students end up in taking jobs in the Computer Science Engineering domain. We use upsampling to balance our dataset for ML analysis. We see that engineering jobs are more popular than non-engineering jobs among students taking AMCAT. Most of the students taking AMCAT (in this dataset), are pursuing an Engineering degree (Fig B2). Electronics and Computer Engineering are the most popular specialization amongst students/AMCAT test takers, with Information Technology following close behind (Fig B3). The age distribution of students taking AMCAT is almost normal with a slight positive skew, centered around 26 years (Fig B4). We also notice that a large number of students leave their first job within the first year, with another peak at 3 years (Fig B5).

Turning our attention to the distribution of other variables when compared to Domains, we draw the following conclusions:

Test Score Type	**Highest scoring students go into these domains**	**Lowest scoring students go into these domains**	**ANOVA Results**
College GPA (Fig B7)	Systems Engineering Research	Support Management and Related	$P < 0.01$ $\omega^2 = 0.03$

	Computer Science Engineering		Med Effect
AMCAT Quant (Fig B8)	Systems Engineering Computer Science Engineering	Support Education	$P < 0.01$ $\omega^2 = 0.02$ Small Effect
AMCAT English (Fig B9)	Systems Engineering Data Science and Engineering Management and Related	Education Electrical and Comm. Engineering	$P < 0.01$ $\omega^2 = 0.02$ Small Effect
AMCAT Domain (Fig B10)	Systems Engineering Computer Science Engineering Research	Electrical and Comm. Engineering Support Management and Related	$P < 0.01$ $\omega^2 = 0.03$ Med Effect

From the data in the table above, we conclude that there is a relationship between scores and domains. Additionally, students taking up jobs in Systems Engineering domain earn the highest salaries on average, followed by Data Science Engineering and Computer Science Engineering domains. Jobs in Education and Support domains pay the least (Fig B6).

Modeling Approach

In our attempt to address our problem statement, we divided our work into two major parts and created separate models.

Identifying Matching Job Domains

As mentioned in the section on data pre-processing, we created job domains from the existing designations in our dataset. Initially, we ended up creating 14 domains which we felt covered all the designations in an exhaustive manner. The split of students in our data based on domains is shown in figure 1 in appendix B.

Minority Class Upsampling

Once we created these domains, we saw that there was a lot of class imbalance as the majority class heavily dominated our dataset. In order to compensate for this imbalance, we decided to upsample the data so that all our classes would have equal representation in the dataset. This upsampling was done on the training set we obtained from a 80-20 split of our dataset. After upsampling, we had ~16,000 records in our up-sampled training data. The test data had ~800 records as a result of the 80-20 split of our main dataset.

Hyper-parameter Tuning and Machine Learning Techniques

After obtaining the up-sampled training data and the test data from the 80-20 split, we decided to tune the hyper-parameters for Logistic Regression, Linear_SVC and Tree-based Classifiers. We chose these machine learning techniques because the task of identifying job domains is a classification problem. Hyper-parameter tuning was done using GridSearchCV and the results obtained from this were used for fitting our up-sampled training data & labels and for predicting the test labels. The process of upsampling, hyperparameter tuning and applying machine learning techniques was done in 2 iterations.

Iteration 1

In the first iteration we worked with the 14 domains that we had created. Among the 14 domains, the majority class heavily dominated our dataset (over 60%). After upsampling, tuning the hyper-parameters for the above mentioned machine learning techniques and applying those techniques to predict test data accuracy, we found our results (accuracy, precision, recall, AUC) to be extremely low for all the machine learning techniques. This was when we decided to refine the domains we created in an attempt to reduce the bias we had created with the 14 domains. This was done in the next iteration.

Iteration 2

After spending some more time analyzing the designations from the original dataset and the domains we had initially created, we refined our domains to come up with 11 domains which had a better split as compared to the first iteration while continuing to remain exhaustive and meaningful. With this 'refined' data, we again performed an 80-20 split and upsampled the training data. Since we did not add or remove data points, we had the same number of records in the training and test data as we had before with the 14 domains. The split of students in our data based on the 11 domains after the second iteration is shown in figure 2 in appendix B.

After performing hyper-parameter tuning for logistic regression (as one-vs-rest classifier) and Linear_SVC, we applied these 2 algorithms to fit our up-sampled training data and their respective labels (with the 11 refined domains). The accuracy, precision, recall and area under curve were very poor (closer to 50%) even after using the tuned hyper-parameters. We then decided to go with tree-based classifiers as we felt that they could give us improved results because they are inherently multi-class. We used ExtraTreeClassifier, DecisionTreeClassifer and RandomForestClassifier. Our results with these tree-based classifiers were an improvement over the results from logistic regression and Linear_SVC.

Providing Better Insights on Salaries

The 'Salary' field in our dataset contains a large range of values. The below tables provides summary statistics for the same field.

Metric	**Value**
Minimum	₹ 35,000.0
Maximum	₹ 4,000,000.0
Mean	₹ 305,686.8
Median	₹ 300,000.0
Standard Deviation	₹ 206,294.7

We also noticed that a number of salary values were less than ₹ 100,000 which is an extremely low salary figure. After closely examining these records, we concluded that it could very well be the case that such responses indicate monthly salaries and not annual figures. But since there was no way of telling the truth, we decided to drop records with salary values less than ₹ 100,000.

Unsupervised Learning Methods

In order to find the perfect buckets to split the salary values, we took help of an unsupervised learning method; K-Means Clustering. Our analysis to find the best of value of *k* resulted in two different values of *k* = [2, 4]. Even though the silhouette coefficient obtained for *k = 2* was of a higher value, we further looked at the boundaries of the different clusters and the number of records in each cluster for both values of *k*. For *k = 2*, we noticed that more than 95% of our dataset was part of just one cluster; this is not something that we want to consider as it would make our model and training set highly biased. Using *k = 4* gave us a significantly better split of data which avoided the bias that *k = 2* might have introduced.

Minority Class Upsampling

After assigning our input data to the created *k = 4* clusters, we split the dataset into Training and Test datasets and upsampled the training dataset to balance the number of records in the majority class and to ensure that we capture enough data points for our minority classes.

Supervised Learning Models and Hyper-parameter Tuning

Our exploration of supervised machine learning models that work best with our data and set of features led us to the conclusion that Decision Trees and Random Forests provide the best performance. Using GridSearch, we tried to find the best hyper-parameter setting for our Decision Trees and Random Forest models. In search for the best hyper-parameter, we searched through a large number of hyper-parameter combinations and proceeded further with fitting newly tuned model with the training data. Unlike the first problem that we are addressing, we have only 4 target classes (or salary buckets). We moved ahead with a 66-34 split on the dataset for training and test subsets for the models that we implemented.

We discuss the results and conclusions from our above models in the next section.

Results

Feature Importance

Apart from the methods and analysis we have talked about above, we also analyzed the importance of the features in our data. Based on our results, we found that the 5 most important features were:

1. 10percentage
2. SecScore
3. 12percentage
4. collegeGPA
5. English

The least important features according to our analysis were:

1. M.Sc. (Tech.)
2. Chemical Engineering
3. Other Specialization
4. Biomedical Engineering
5. Civil Engineering

Identifying Matching Job Domains

Below are the results for the tree-based classification models that we trained after tuning the hyper-parameters using GridSearch.

Learning Model	**Hyper-parameters**	**Training Accuracy**	**Test Accuracy**
Decision Trees	Criterion = Entropy Max Depth = 20 Max Features = 7 Min Samples Leaf = 3 Min Samples Split = 2	**80.95**	**79.94%**
Extra Tree Classifier	Criterion = Entropy Max Depth = 20 Max Features = 10 Min Samples Leaf = 3 Min Samples Split = 2	**74.3%**	**65.16%**
Random Forest	Criterion = Entropy Max Depth = 20 Max Features = 5 Min Samples Leaf = 3 Min Samples Split = 7 Bootstrap = False	**88.79%**	**83.71%**

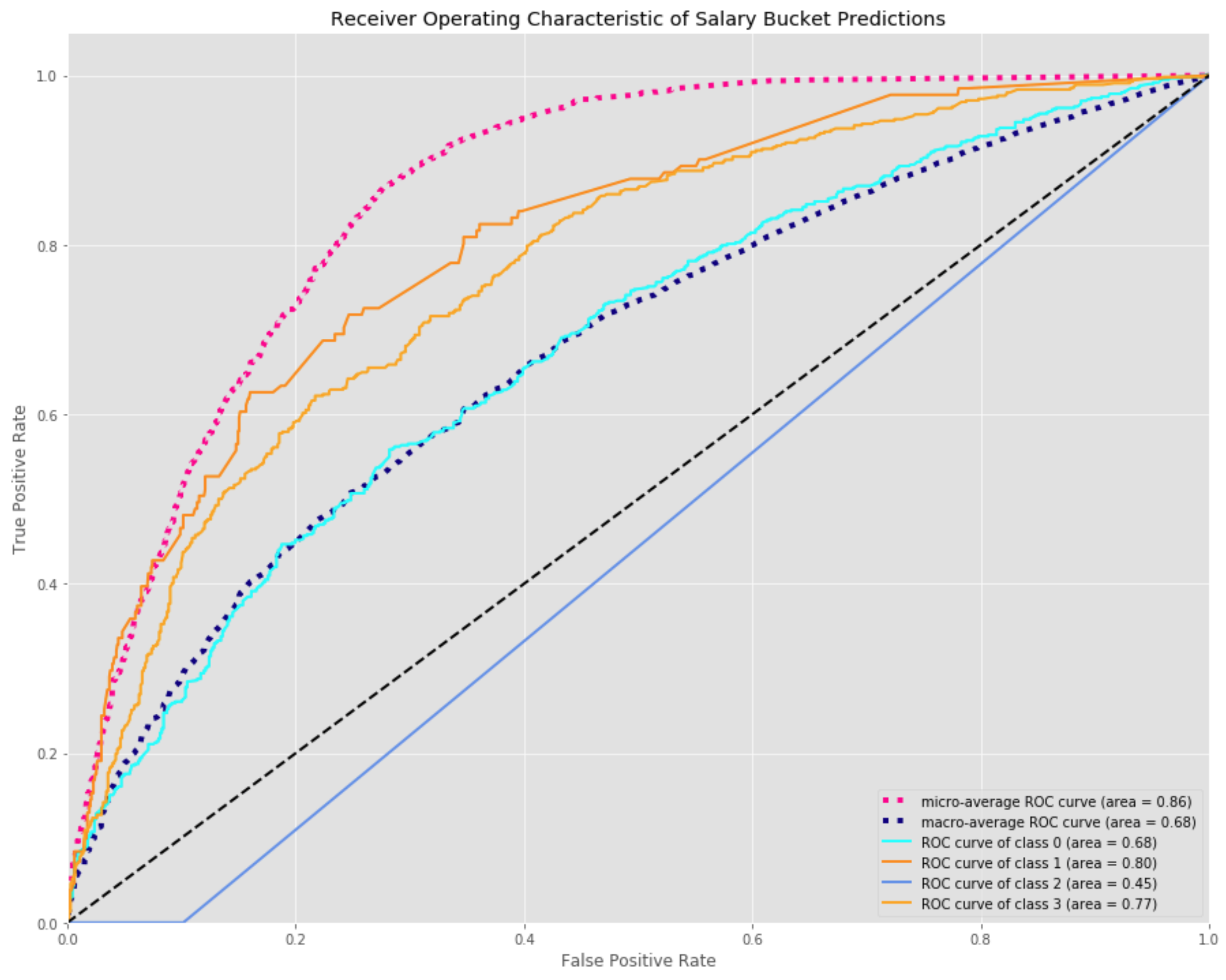
For the above classifiers, the confusion matrices and graphs showing area under curve are attached in Appendix B.

Providing Better Insights on Salaries

Below are the results for the supervised learning models that we trained after tuning the hyper-parameters using GridSearch.

Learning Model	**Hyper-parameters**	**Training Accuracy**	**Test Accuracy**
Decision Trees	Max Depth = 20 Criterion = 'gini' Max Features = 20 Min Samples Leaf = 1 Min Samples Split = 2	**82.37%**	**50.90%**
Random Forests	# Estimators = 25 Max Depth = 20 Criterion = 'entropy' Max Features = 20 Min Samples Leaf = 1 Min Samples Split = 5	**88.30%**	**60.40%**

From the performance of our two tuned models, we can see that there was a significant improvement in test set accuracy from Decision Trees to Random Forests. Looking below at the AUC plot for Random Forests, we can observe that the model is performing very poorly for a specific class and this might explain the overall poor accuracy of the test set. Nevertheless, it is a significant improvement over Decision Trees.

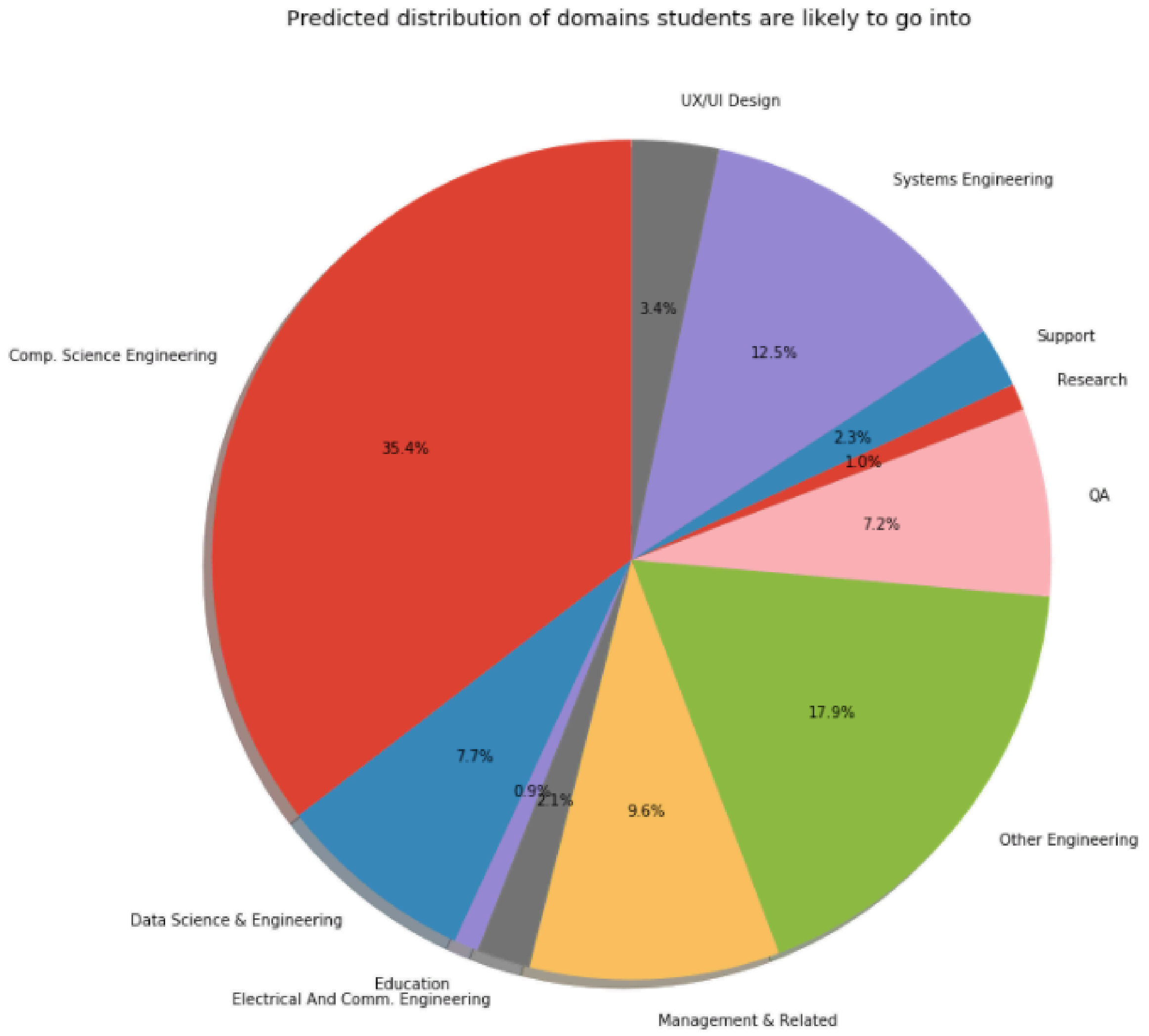


Discussion & Conclusion

The results of our feature importances reconcile with our personal experiences. If we look at the 5 most important features, we observe that these features are grades and scores. Most companies visiting campuses for placements set cut-off limits for the GPA as well as the percentages obtained in the 10th & 12th grades. SecScore (a derived feature) and English are performance indicators of a student on the AMCAT exam and are important for finding a student's knowledge and aptitude.

If we consider the least important features, we see that a majority of them are specializations (or the major) of students. Since the career interests of students often change during the course of their studies, quite a few of them end up taking jobs that are not related to their specializations. Because placement cells often fail to take this factor into account, they end up inviting the wrong mix of companies based on the number of students in every specialization. Our model can help them rectify this approach by giving them a holistic view of student profiles.

If we look at the results of the classifiers used for predicting the job domains, we find that the Random Forests performs the best. This can be verified by comparing the ROC graph and confusion matrix of the classifier with the ROC graphs and confusion matrices of the Decision Tree and Extra Tree classifiers. In terms of performance metrics, we got a test accuracy of over 80% with the Random Forest Classifier. On taking a closer look at the ROC graph, we see that the classifier does a better job than the other 2 classifiers in predicting true positives for a greater number of classes. If we compare the 3 confusion matrices, we find that the Random Forest Classifier has a lesser misclassification rate than the other 2 classifiers.



Pie chart showing the domain percentage as predicted by the Random Forest Classifier

The above pie-chart gives us a very good idea of the job domains that the students in our test set are likely to enter into. If this information is available to the placement cell of a college or university, they would be able to make an informed decision about the types of companies that they should target to come on campus to hire students. If the placement cell were to focus only on inviting companies in the Computer Science & Engineering and System Engineering domains, they would not be able to match a majority of the students with various appropriate job domains that are suitable for them. In conclusion, a model like the one we have created using the Random Forest Classifier will help placement cells target the right set of companies and help students in finding the right job based on their interests and academic profile.

Further Improvements and Future Scope

We believe that a number of small improvements in the data collection process can drastically improve the performance of the underlying models. One of the major challenges that we faced with this dataset was that of free-form text responses. Multiple respondents with similar designations had different values in the designation column as there was no data quality check in place when the data was collected. This is what motivated us to group designations into different domains and at the same time preserve the meaning. Salary was another column that needed to be checked against monthly or annual income figures. Further, if instead of providing exact salary figures, Aspiring Minds can aim to collect salary brackets for the different respondents. If enough data points, this would eliminate the need for using unsupervised models such as K-Means.

Appendix A: Aspiring Minds Data Challenge Winning Submissions

1. Predict job success based on Student's credentials - http://research.aspiringminds.com/wp-content/uploads/2016/06/TEAM-DS-Final_report.pdf
2. Understanding Labor Markets - http://research.aspiringminds.com/wp-content/uploads/2016/06/Paper13_CameraReady.pdf
3. Understanding the Indian Labour Market: A Data Centric Approach - http://research.aspiringminds.com/wp-content/uploads/2016/06/Paper29_CameraReady.pdf
4. Bayesian Visual Analysis of the Indian Labour Market - http://research.aspiringminds.com/wp-content/uploads/2016/06/Paper11_CameraReady.pdf

Appendix B: Exploratory Data Analysis Graphs

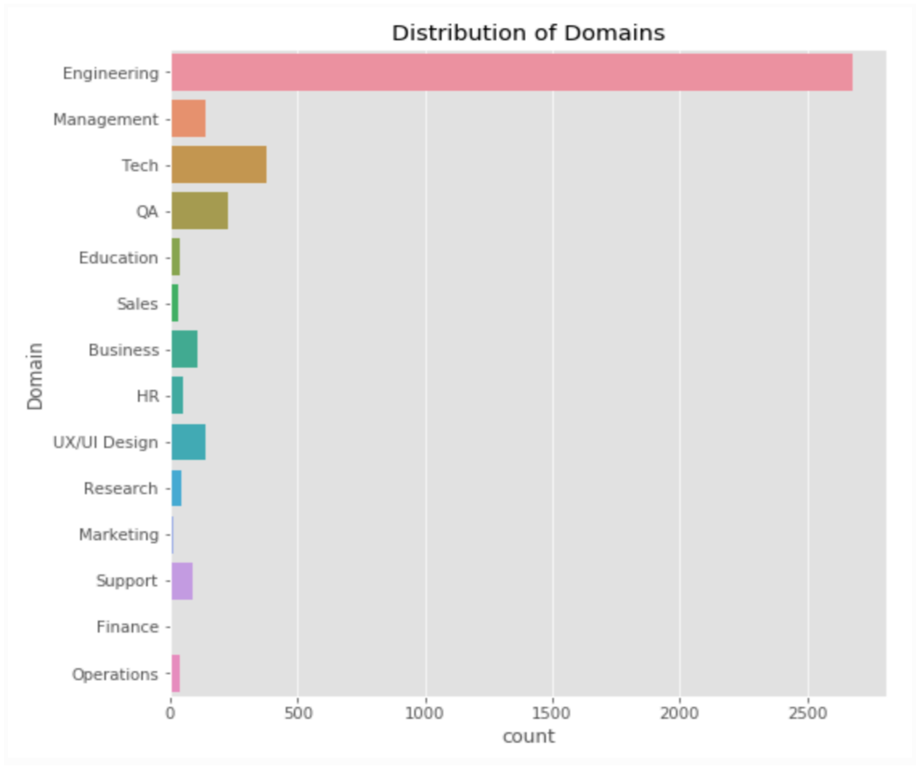


Fig B1a. Distribution of Domains in original dataset

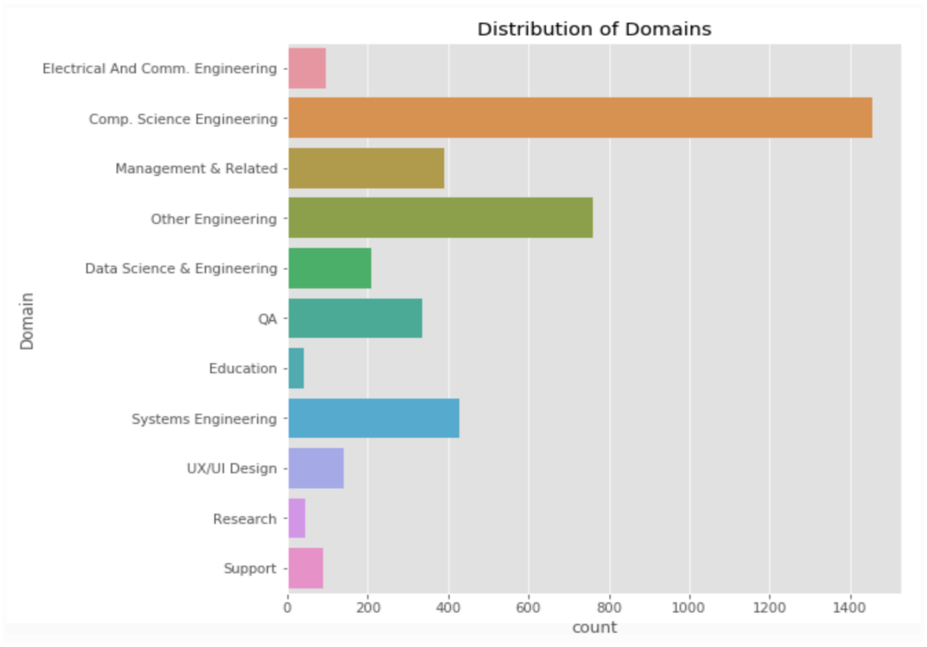


Fig B1b. Distribution of Domains after second iteration

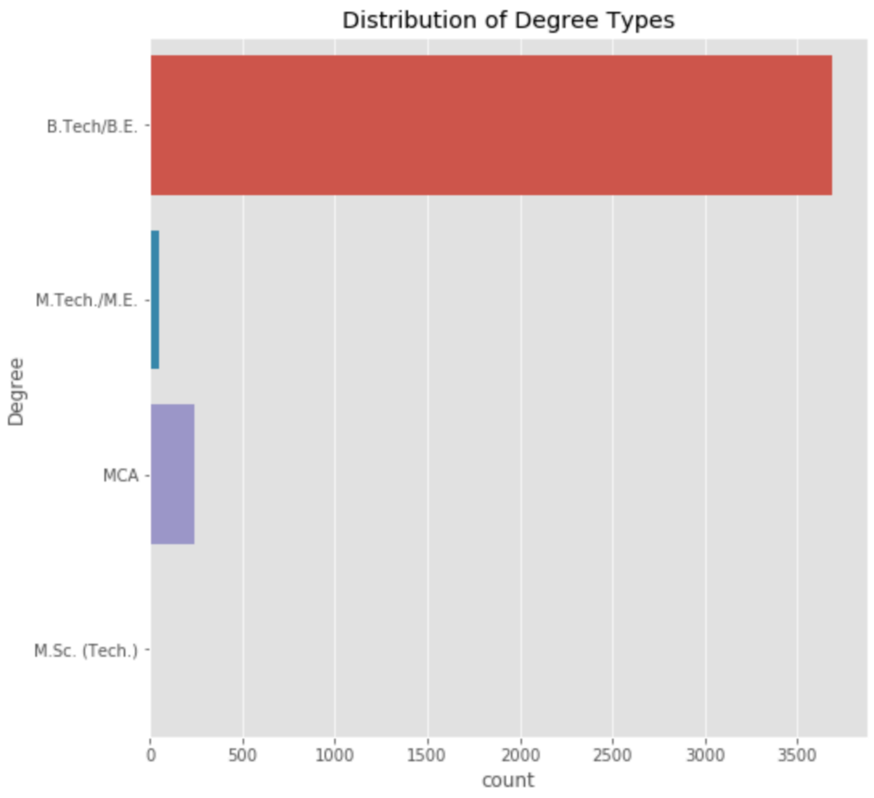


Fig B2. Distribution of Degree Types

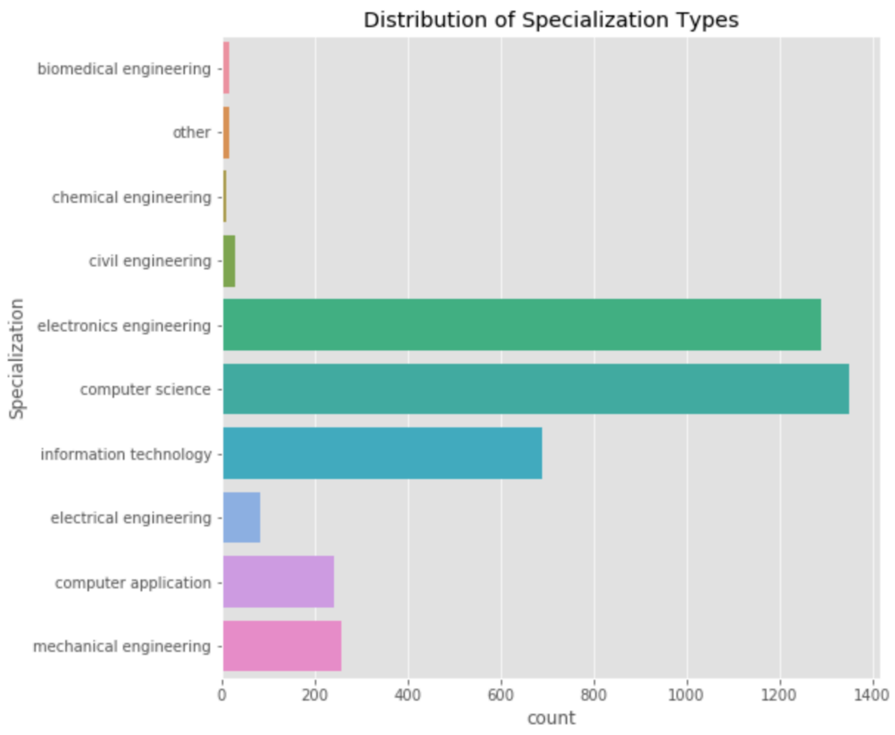


Fig B3. Distribution of Specialization Types

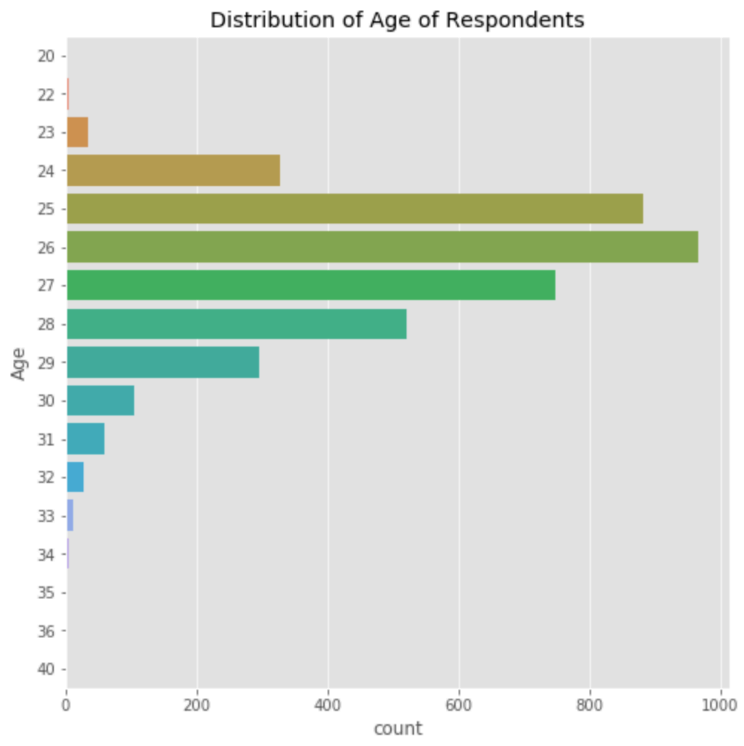


Fig B4. Distribution of Age of students surveyed

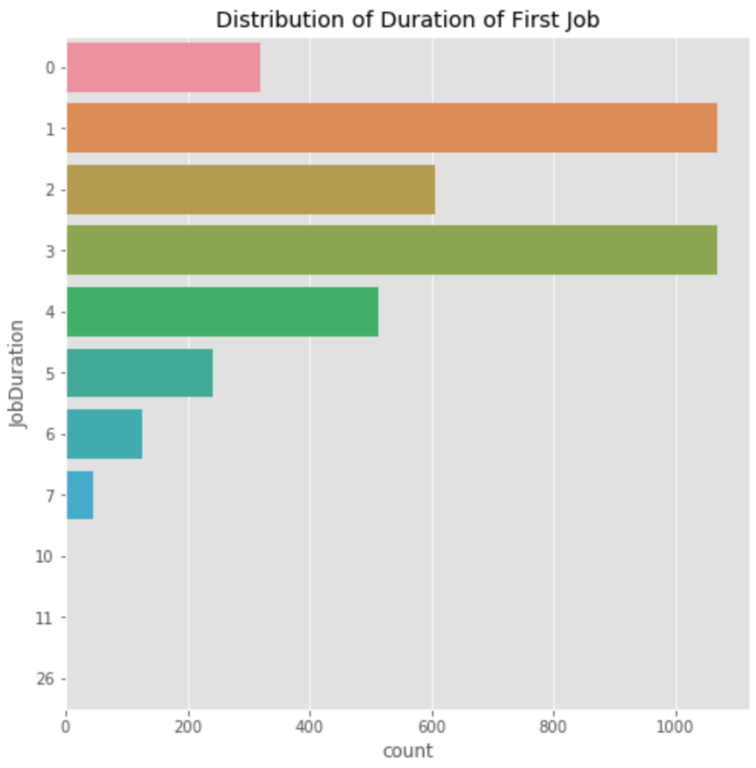


Fig B5. Distribution of Duration of First Job

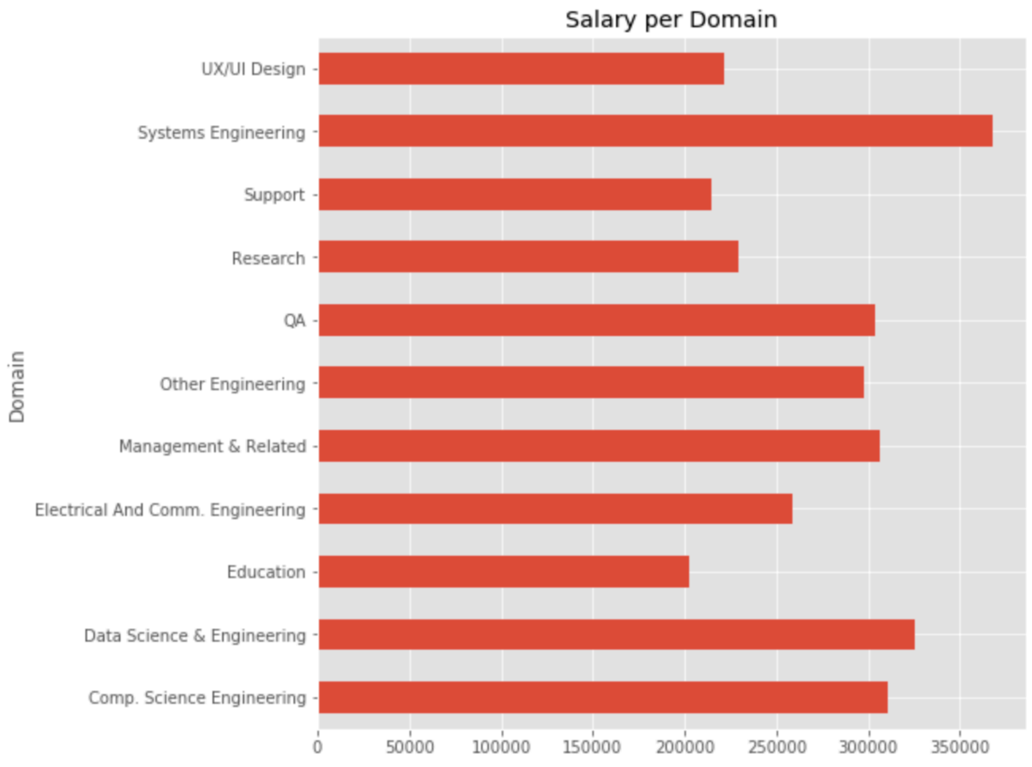


Fig B6. Distribution of Salary per Domain

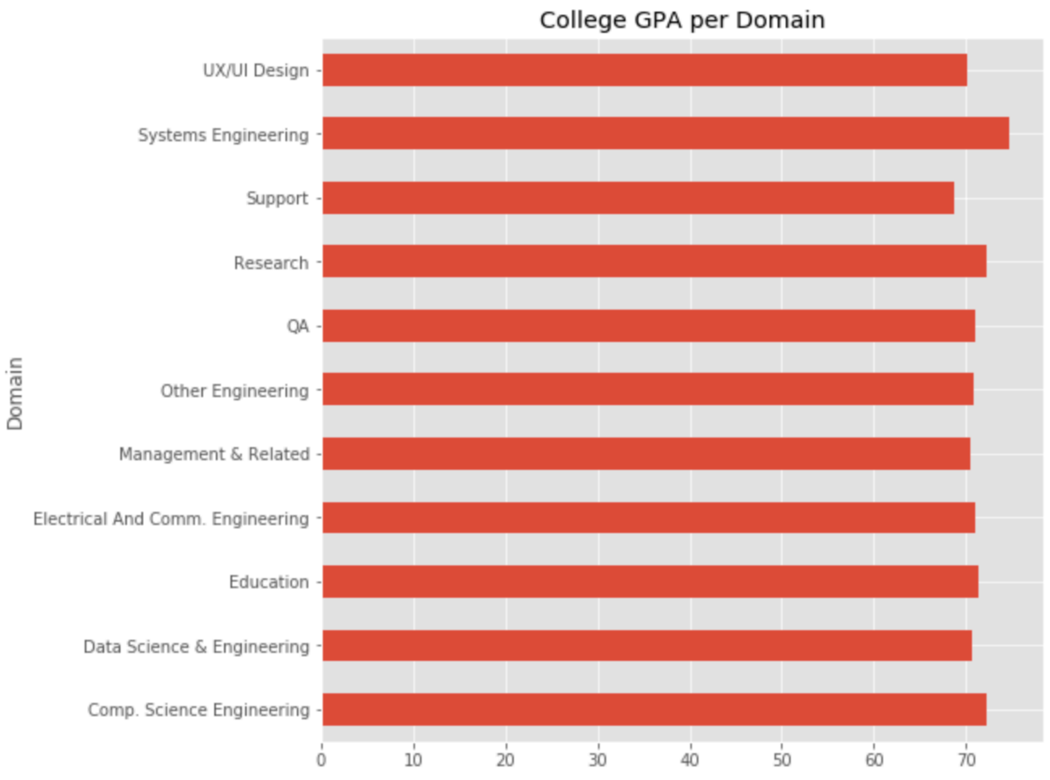


Fig B7. Distribution of College GPA per Domain

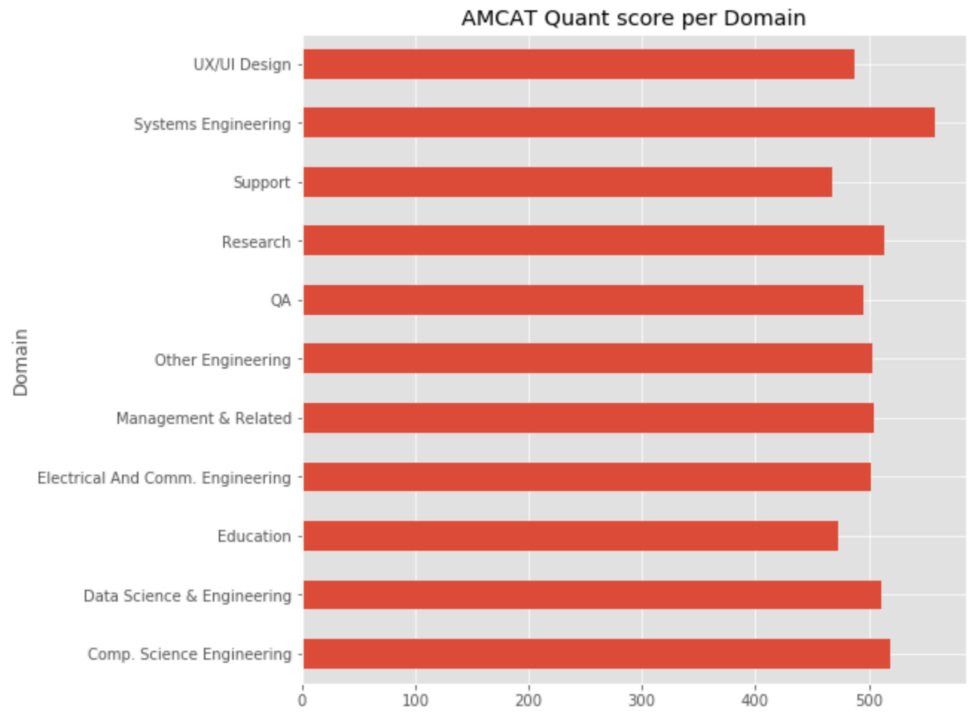


Fig B8. Distribution of AMCAT Quant scores per Domain

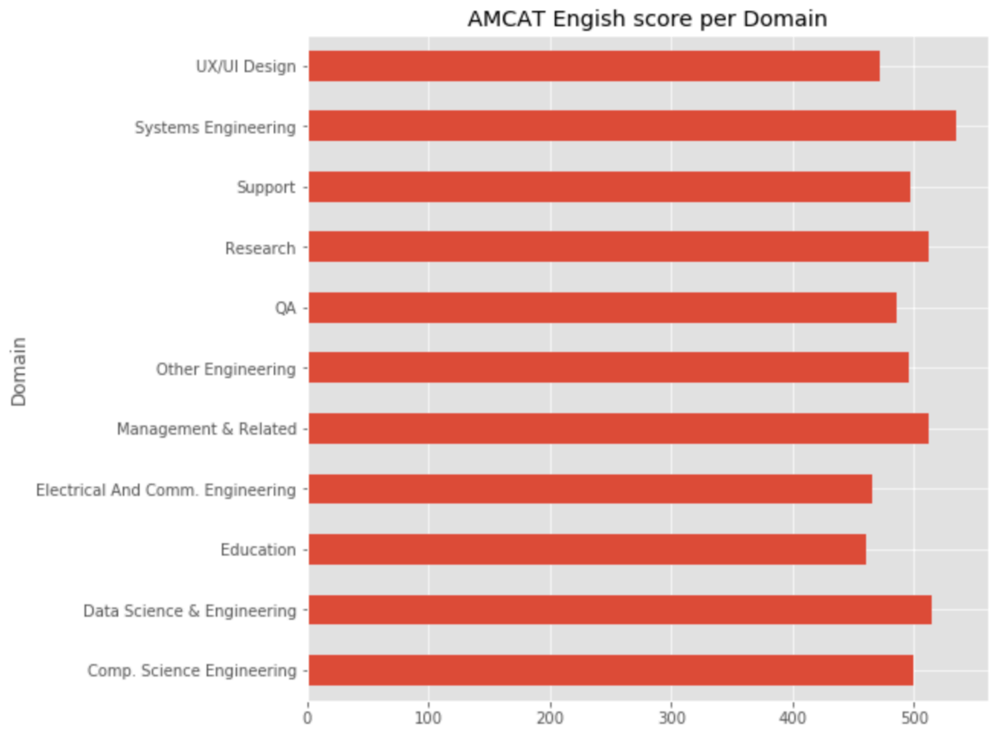


Fig B9. Distribution of AMCAT English scores per Domain

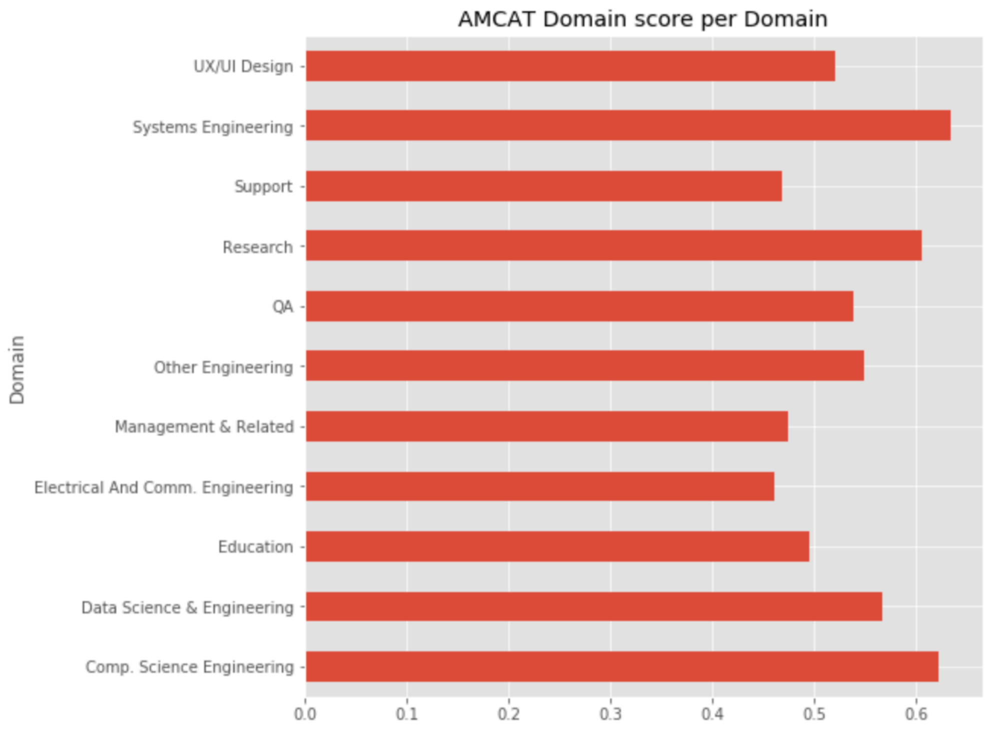
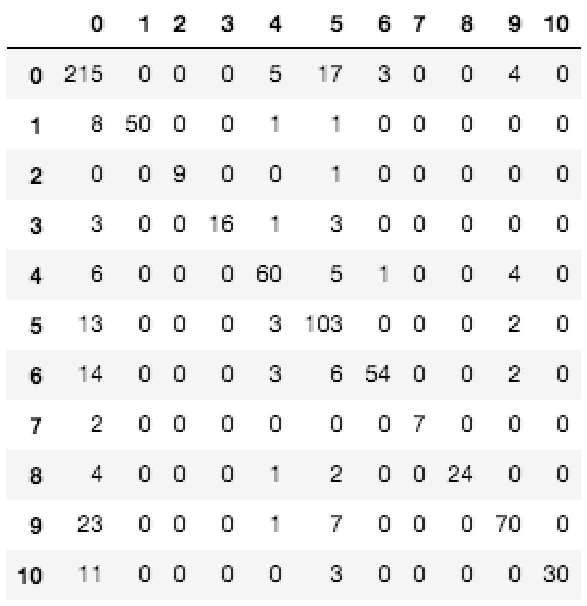
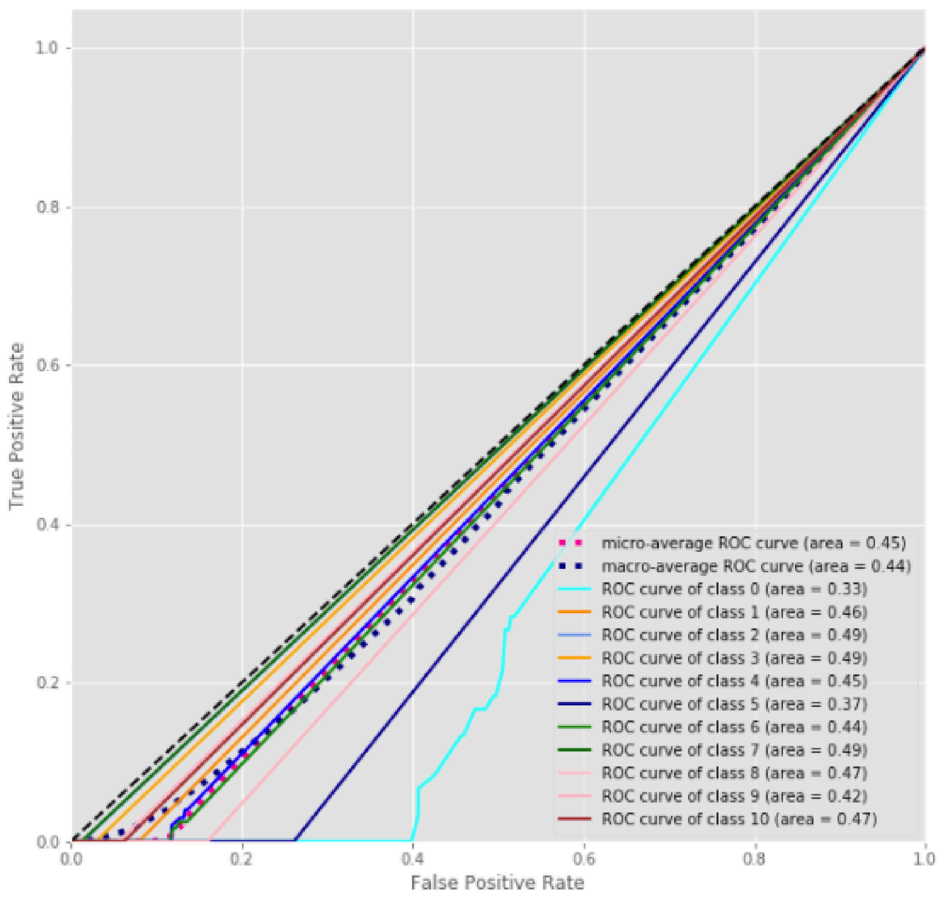


Fig B10. Distribution of AMCAT Domain Scores per Domain

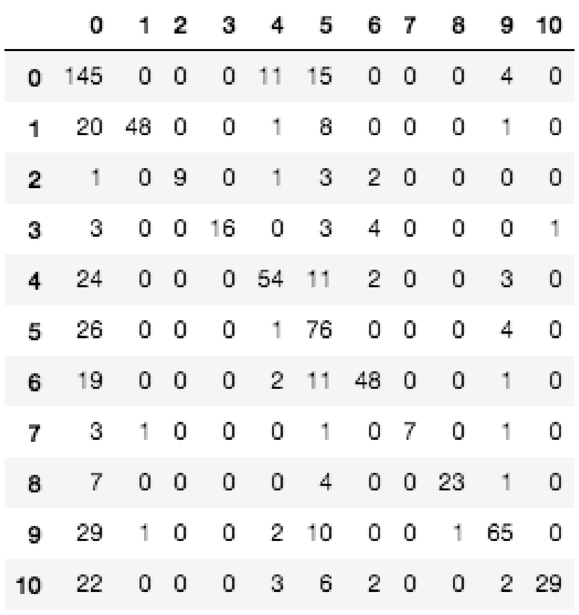


	0	1	2	3	4	5	6	7	8	9	10
0	215	0	0	0	5	17	3	0	0	4	0
1	8	50	0	0	1	1	0	0	0	0	0
2	0	0	9	0	0	1	0	0	0	0	0
3	3	0	0	16	1	3	0	0	0	0	0
4	6	0	0	0	60	5	1	0	0	4	0
5	13	0	0	0	3	103	0	0	0	2	0
6	14	0	0	0	3	6	54	0	0	2	0
7	2	0	0	0	0	0	0	7	0	0	0
8	4	0	0	0	1	2	0	0	24	0	0
9	23	0	0	0	1	7	0	0	0	70	0
10	11	0	0	0	0	3	0	0	0	0	30

Confusion Matrix for Decision Tree Classifier

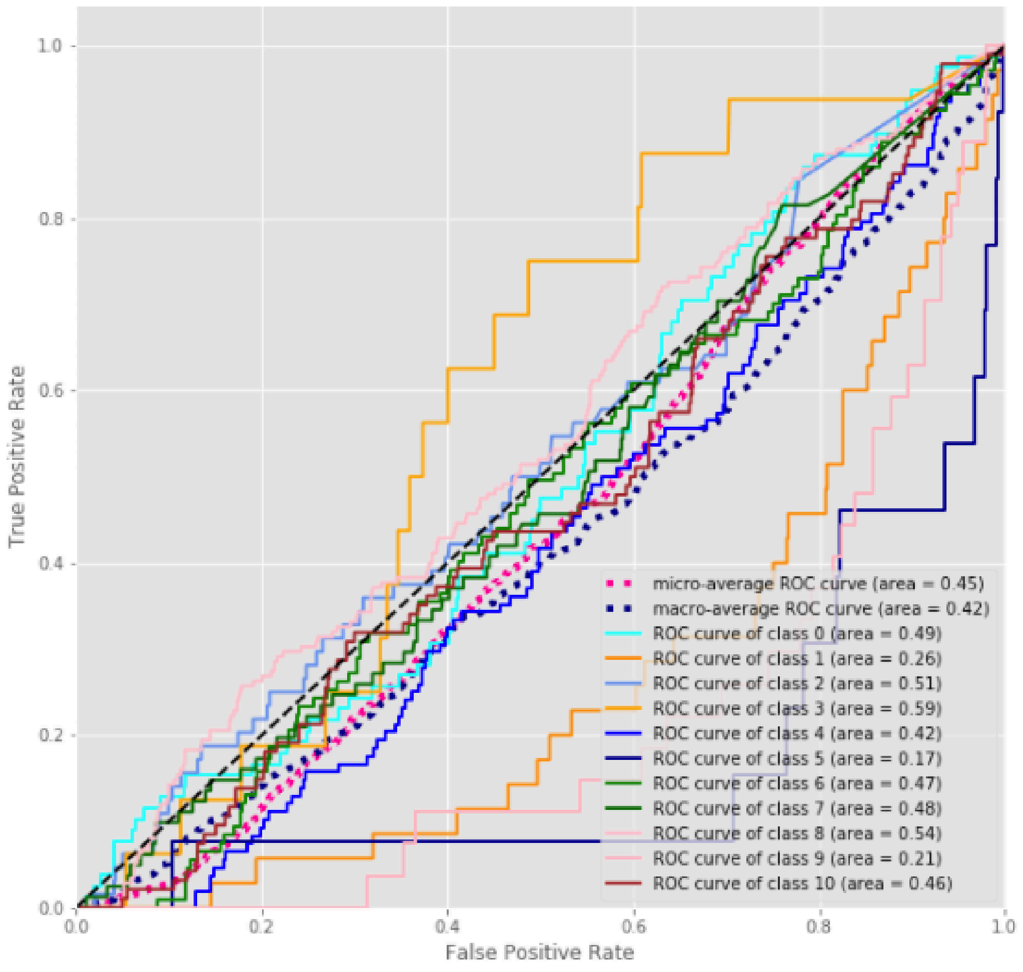


Receiver Operator Characteristics Curve for Decision Tree Classifier

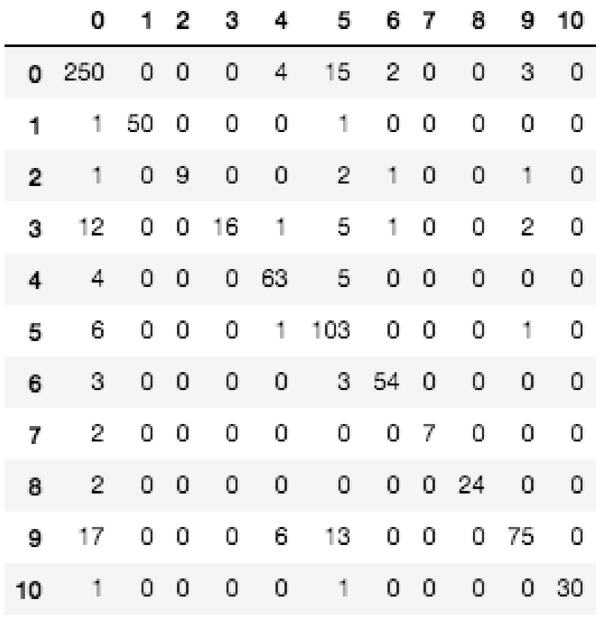


	0	1	2	3	4	5	6	7	8	9	10
0	145	0	0	0	11	15	0	0	0	4	0
1	20	48	0	0	1	8	0	0	0	1	0
2	1	0	9	0	1	3	2	0	0	0	0
3	3	0	0	16	0	3	4	0	0	0	1
4	24	0	0	0	54	11	2	0	0	3	0
5	26	0	0	0	1	76	0	0	0	4	0
6	19	0	0	0	2	11	48	0	0	1	0
7	3	1	0	0	0	1	0	7	0	1	0
8	7	0	0	0	0	4	0	0	23	1	0
9	29	1	0	0	2	10	0	0	1	65	0
10	22	0	0	0	3	6	2	0	0	2	29

Confusion Matrix for Extra Tree Classifier

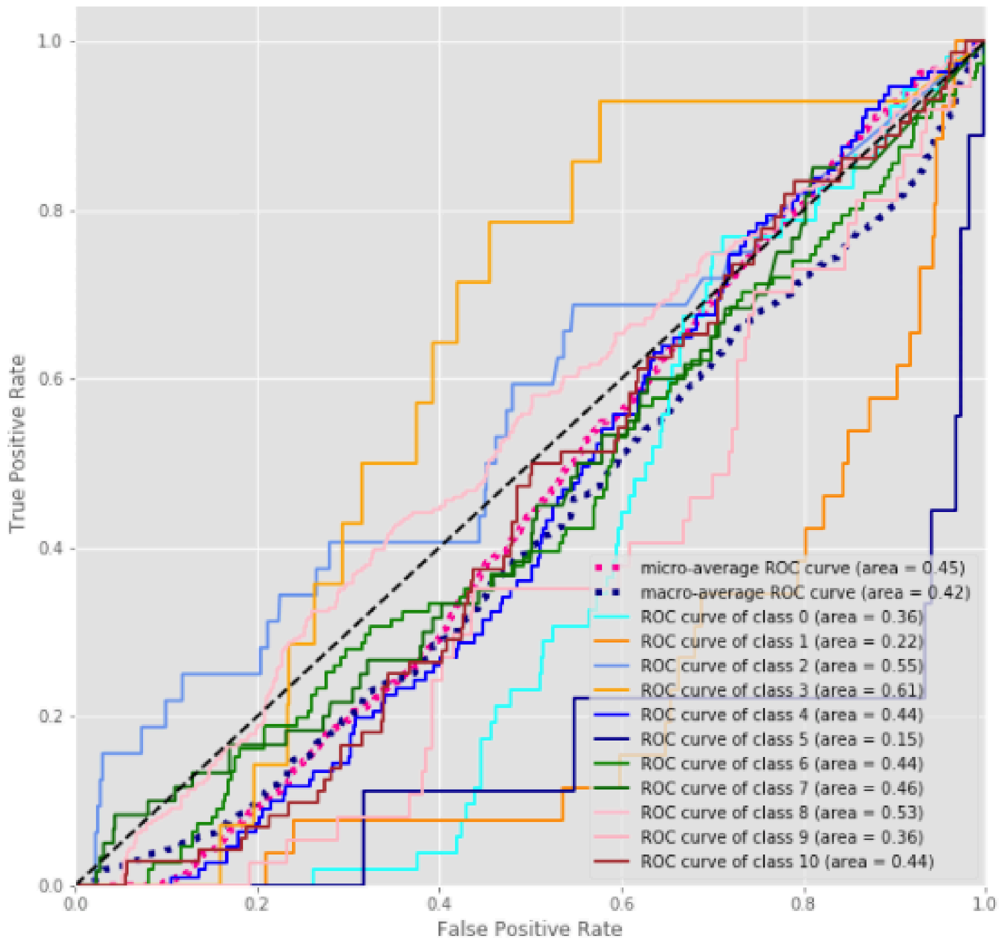


Receiver Operator Characteristics Curve for Extra Tree Classifier

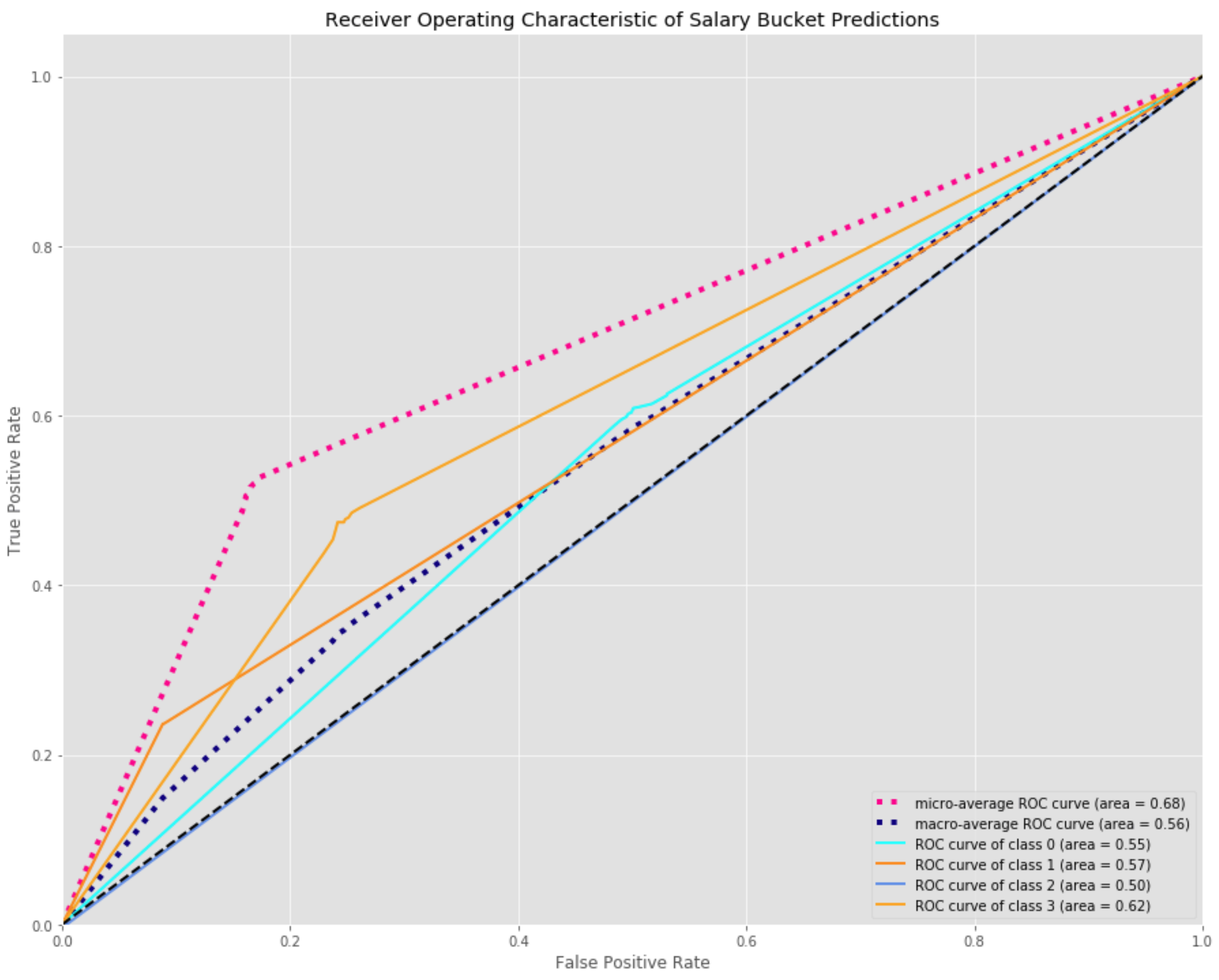


	0	1	2	3	4	5	6	7	8	9	10
0	250	0	0	0	4	15	2	0	0	3	0
1	1	50	0	0	0	1	0	0	0	0	0
2	1	0	9	0	0	2	1	0	0	1	0
3	12	0	0	16	1	5	1	0	0	2	0
4	4	0	0	0	63	5	0	0	0	0	0
5	6	0	0	0	1	103	0	0	0	1	0
6	3	0	0	0	0	3	54	0	0	0	0
7	2	0	0	0	0	0	0	7	0	0	0
8	2	0	0	0	0	0	0	0	24	0	0
9	17	0	0	0	6	13	0	0	0	75	0
10	1	0	0	0	0	1	0	0	0	0	30

Confusion Matrix for Random Forest Classifier



Receiver Operator Characteristics Curve for Random Forest Classifier


Receiver Operating Characteristic of Salary Bucket Predictions
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.68)
macro-average ROC curve (area = 0.56)
ROC curve of class 0 (area = 0.55)
ROC curve of class 1 (area = 0.57)
ROC curve of class 2 (area = 0.50)
ROC curve of class 3 (area = 0.62)

2) Final report (2400-3000 words): The final report will be the primary basis for evaluation of your project. The first page of your report should consist of a title page that lists the names of all group members, a title for your project, and a short abstract, no more than 250 words, which describes the key findings of your group. Include in your report any figures, tables, or multimedia (as hyperlinks) that you develop in the course of your project. Code and other relevant but non-essential material may optionally be included as appendices. The content of the title page, references, footnotes, and appendices does not count toward the word limit. In the body of your report, you should include a discussion of the following:

- Datasets used
- Primary methods implemented
- Results
- Discussion
- Conclusions
- References

Your project will be evaluated, and your grade determined, using the following criteria:

1. Relevance to class: The project must engage with supervised learning methods, and those methods must be necessary and not incidental to the execution of your project
2. Novelty and usefulness: You should be able to convince your grandma that your project is interesting, and convince a skeptic that no one has done this before. Summarize prior work you have seen on this or closely-related topics.
3. Methodological rigor: Choose data sets with enough examples to get statistically significant results; **use appropriate methods in appropriate ways; don't ask unanswerable questions; design your experiments correctly (with separate test data, cross-validation, and reasonable baselines); interpret your results correctly.**
4. Clarity of written report -- Clearly and succinctly articulate your question and/goals early. Clearly describe how you process and analyze the data, and summarize the data being used in the analysis. Add figures and tables and other effective visualizations.

Rubric

Criteria	**Pts**

Description of prior work	5.0 pts
Methodological rigor - are modeling/analysis decisions clearly justified?	10.0 pts
Description of data and description of analysis performed - is everything crystal clear?	8.0 pts
Effectiveness of figures and tables	7.0 pts
Presentation and interpretation of results - are conclusions clearly stated and justified?	10.0 pts
Writing quality and professionalism/copyediting	5.0 pts
Ambition and novelty	5.0 pts
Quality of post on Online Gallery	5.0 pts
	Total Points: 55.0

Josh’s Comments:

Your team did a good job presenting a relatively complex project. That said, I have several suggestions for improvement:

- The questions you ask on the first slide are interesting, but some of them are not questions that can really be answered with your data. More importantly, it's not clear what your answer is to several of these questions, from the presentation. Make sure to connect these dots clearly in your report. Eliminate questions that cannot be answered.
- The slides themselves were very cluttered and the font on almost all of the figures was illegible. When crafting presentations, you need to put yourselves in the perspective the person attending the presentation, not the person giving the presentation. From that perspective, do the figures make sense? Please make sure to clean these up and make them much more professional in your report.
- It was never quite clear how a "placement cell" was defined