

Intro to Machine Learning Applications
MGMT6560 Spring 2020

Amazon.com - Employee Access Challenge
May 4, 2020

Submitted to:

Professor Jason Kuruzovich

Submitted By:

Vrishti Jain

Content

1. Executive Summary	2
2. Dataset	3
3. Benchmarking of Other Solution	4
4. Data description and Initial Processing	6
5. Modeling	10
6. Summary and Conclusion	15
7. Appendices	17

Section 1: Executive Summary

Problem: To take data about employee's role information and a resource code and will return whether or not access should be granted or not.

The dataset consists of real historical data collected from 2010 & 2011. Each employee has certain information associated with them which is the criteria to decide whether a particular privilege to the employee will be given or not.

When an employee starts work at any organization, they first need to get access to the computer necessary to fulfill their task. Such access can allow an employee to read/manipulate information via various web portals or applications. Employees who perform the duties of a given role are believed to have access to the same or similar resources. A professional supervisor then takes time to manually assign the required access to address obstacles to entry. When workers transfer around an organization, this discovery/recovery process of access is wasting a nontrivial amount of time and resources.

The dataset was purely categorical. All the fields were given ID for the class they belonged to, so even though the fields had numeric values, they represented the class they belonged to. The main aim was to find the variables that were important with respect to the target variable. With the help of logistic regression classifier, 3 variables looked important which were ROLE_ROLLUP_1, ROLE_ROLLUP_2 and ROLE_CODE. Although with Extra Tree classifier important features were RESOURCE, MGR_ID and ROLE_DEPTNAME.

Since the target variable is categorical 0 or 1, I tried classification algorithms such as Logistic Regression classifier, Extra Tree classifier and Cat Booster algorithms. Additionally the features are One hot encoded.

The results were different with One hot encoded Dataset and the origin one. Also the accuracy was different for different algorithms. Feature selection with respect to algorithms doesn't improve accuracy over all. Additionally, in case of Extra Tree Classifier, it dropped the roc score.

Dataset:

train.csv - The training set. Each row has the ACTION (ground truth), RESOURCE, and information about the employee's role at the time of approval

test.csv - The test set for which predictions should be made. Each row asks whether an employee having the listed characteristics should have access to the listed resource.

The columns present in the dataset:

- **ACTION**: ACTION is 1 if the resource was approved, 0 if the resource was not
- **RESOURCE**: An ID for each resource.
- **MGR_ID**: The EMPLOYEE ID of the manager of the current EMPLOYEE ID record; an employee may have only one manager at a time.
- **ROLE_ROLLUP_1**: Company role grouping category id 1 (e.g. US Engineering). These are all numeric values.
- **ROLE_ROLLUP_2**: Company role grouping category id 2 (e.g. US Retail). These are also numeric values.
- **ROLE_DEPTNAME**: Company role department description (e.g. Retail). These are also numeric values.
- **ROLE_TITLE**: Company role business title description (e.g. Senior Engineering Retail Manager). Also numeric values.
- **ROLE_FAMILY_DESC**: Company role family extended description (e.g. Retail Manager, Software Engineering)
- **ROLE_FAMILY**: Company role family description (e.g. Retail Manager)
- **ROLE_CODE**: Company role code; this code is unique to each role (e.g. Manager)

Since the task is of classification whether a particular resource will be given to the employee or not, classification algorithms such as Logistic Regression and other Tree-based classification models can be applied to the dataset.

Section 2: Benchmarking of Other Solution

Notebook Name	Feature Approach	Model Approach	Train/Test Performance
KaggleDays SF: 1. Amazon - Baseline	Removed one feature as it had one to one relation with other features. All the features are categorical variables, features are transformed using one-hot encoding.	After one hot encoding the variables, they are fit into the Logistic Regression model and its performance is checked using the 5 fold cross-validation.	AUC score of 0.86 on test data and AUC score of 0.97 on train data.
KaggleDays SF: 2. Amazon - Unsupervised encoding	In this notebook, they tried different encoding techniques and then applying models to see the result. Applied Label encoding on all the attributes which resulted in a total of 40 features. Then tried the one-hot encoding, which resulted in a total of 16600 features which created a very sparse dataset. I also tried SVD encoding which transforms and reduces the matrix to a vector. This also resulted in a total of 56 features. The notebook also tried frequency encoding which focuses on how many times a particular value is uniquely present and assigns that as a value. This doesn't increase the number of attributes.	On the label encoded data, one hot encoded dataset, SVD encoded dataset and frequency encoded dataset, the ExtraTreesClassifier model and cross-validation were applied.	With label encoding, on test data: 0.86 and on train data: 1 One hot encoding, on test data: 0.875 and on train data: 1 With SVD encoding, on test data:0.86 and on train data:0.99. And with Frequency encoding, on test data:0.82 and on train data:0.99. These are the AUC scores.
KaggleDays SF: 3.	In this notebook, they	On these different	With Simple Target

Amazon - Supervised encoding	also tried encoding techniques, starting with Simple Target Encoding one before CV and one inside the CV loop. Then moving ahead with the Target Encoding Smoothing with the dataset mean and level mean. Then making embedding noisy. They also tried to use feature pairs to create a new set of categorical features which resulted in a total of 36 features instead of 8. And on these features, using Target Encoding Expanding Mean and target Encoding Smoothing with CV to create embeddings.	encodings, models were applied along with cross-validating. Extra Trees Classifier model is used after each step.	Encoding without encoding inside the CV:: 0.9749 and within the CV: 0.74. With Target Encoding Smoothing, AUC score of 0.78 is achieved. Now after adding noise: 0.85. With feature pairs and then using the two encoding methods on the feature, AUC score was 0.8973
------------------------------	--	---	--

In all these notebooks, since the dataset consists of categorical attributes, several encoding techniques were applied. All of them have some benefits and some disadvantages as well. With one hot encoding, it is easy to understand and implement. However, it creates a very sparse dataset and along with tree-based models, it takes a lot of time to process and can drop the performance of the model. Also with frequency encoding in which encoded values represent how many times this unique value is presented in your data. It is also easy to understand and implement but it can't be used in case if the categorical values are balanced. While with SVD encoding, it provides a rich multidimensional dataset, though sometimes it is not easy to understand what it really represents. With frequency encoding, the values are randomly assigned so they are unbiased, though it's hard to explain the significance it has on the dataset.

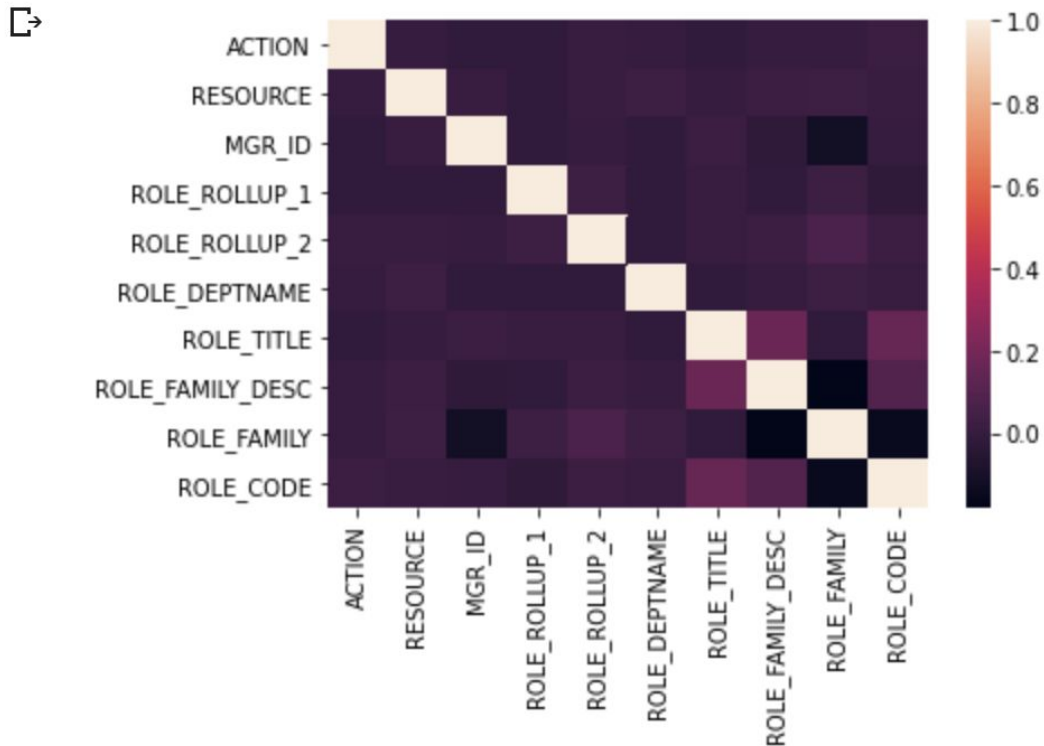
Simple Target Encoding is the simplest way to encode each unique value by target means. It is easy to implement and powerful, though it introduces data leakages. One way to deal with it is to build features using a target always inside the cross-validation loop. Further with the target encoding smoothening, it is to address the problem with low- frequency values. In this step a weighted sum of two means: dataset means and level mean, where level mean is the mean of particular unique value in your feature. It is also easy and robust but it introduces two additional parameters per feature, which could be hard to tune. Further adding noise helps in increasing the accuracy.

Section 3: Data description and Initial Processing

```
[ ] train.head()
# though it dataset has categorical values but all the fields contain numerical categorical data. Like primary key for the given attribute.
```

	ACTION	RESOURCE	MGR_ID	ROLE_ROLLUP_1	ROLE_ROLLUP_2	ROLE_DEPTNAME	ROLE_TITLE	ROLE_FAMILY_DESC	ROLE_FAMILY	ROLE_CODE
0	1	39353	85475	117961	118300	123472	117905	117906	290919	117908
1	1	17183	1540	117961	118343	123125	118536	118536	308574	118539
2	1	36724	14457	118219	118220	117884	117879	267952	19721	117880
3	1	36135	5396	117961	118343	119993	118321	240983	290919	118322
4	1	42680	5905	117929	117930	119569	119323	123932	19793	119325

All the values are numerical but they are actually categorical. As the number just represents the group they belong to. Like manager id is some integer but it is like a unique primary key. And this is true for every value in the dataset.



```
[17] corrMatrix
```

```
↳
```

	ACTION	RESOURCE	MGR_ID	ROLE_ROLLUP_1	ROLE_ROLLUP_2	ROLE_DEPTNAME	ROLE_TITLE	ROLE_FAMILY_DESC	ROLE_FAMILY	ROLE_CODE
ACTION	1.00	0.00	-0.01	-0.01	0.01	0.00	-0.01	0.00	0.00	0.02
RESOURCE	0.00	1.00	0.01	-0.01	0.01	0.03	0.00	0.02	0.03	0.01
MGR_ID	-0.01	0.01	1.00	-0.01	-0.00	-0.01	0.02	-0.02	-0.12	-0.00
ROLE_ROLLUP_1	-0.01	-0.01	-0.01	1.00	0.03	-0.01	0.01	-0.01	0.03	-0.02
ROLE_ROLLUP_2	0.01	0.01	-0.00	0.03	1.00	-0.01	0.01	0.02	0.07	0.02
ROLE_DEPTNAME	0.00	0.03	-0.01	-0.01	-0.01	1.00	-0.01	-0.00	0.03	0.01
ROLE_TITLE	-0.01	0.00	0.02	0.01	0.01	-0.01	1.00	0.17	-0.01	0.16
ROLE_FAMILY_DESC	0.00	0.02	-0.02	-0.01	0.02	-0.00	0.17	1.00	-0.18	0.09
ROLE_FAMILY	0.00	0.03	-0.12	0.03	0.07	0.03	-0.01	-0.18	1.00	-0.15
ROLE_CODE	0.02	0.01	-0.00	-0.02	0.02	0.01	0.16	0.09	-0.15	1.00

Moving onto the correlation between these categorical values. I calculated the correlation value and rounded them to two decimal points. As these numbers represent just the category they belong to and we don't have the exact category name. We can say that, due to the unbiased and unordered sequence of all the different categories in each feature, there is no direct correlation between the numeric value of category and the real category. The way numeric value is assigned to roll rollup 1, lets say US engineering, the numeric value doesn't signify any importance and it only represents a category it belongs to.

Moving on to see unique values in each feature, we can see the number of unique categories are really large in number. Also role code and role title have the same number of unique values.

```
[ ] train.apply(lambda x: len(x.unique()))
# we can see that role_code and role_title has equal number of unique values. It might be the case that both of them are the same.
```

```
↳
```

```
ACTION          2
RESOURCE        7518
MGR_ID          4243
ROLE_ROLLUP_1   128
ROLE_ROLLUP_2   177
ROLE_DEPTNAME    449
ROLE_TITLE       343
ROLE_FAMILY_DESC 2358
ROLE_FAMILY       67
ROLE_CODE        343
dtype: int64
```

Now by grouping the data based on both role code and role title and then grouping only by role code, we can see there is a one to one relationship between these two and one of them can be removed.


```

role_code = 'ROLE_CODE'
role_title = 'ROLE_TITLE'

group_two = train.groupby([role_code,role_title]).size()
group_single = train.groupby([role_code]).size()

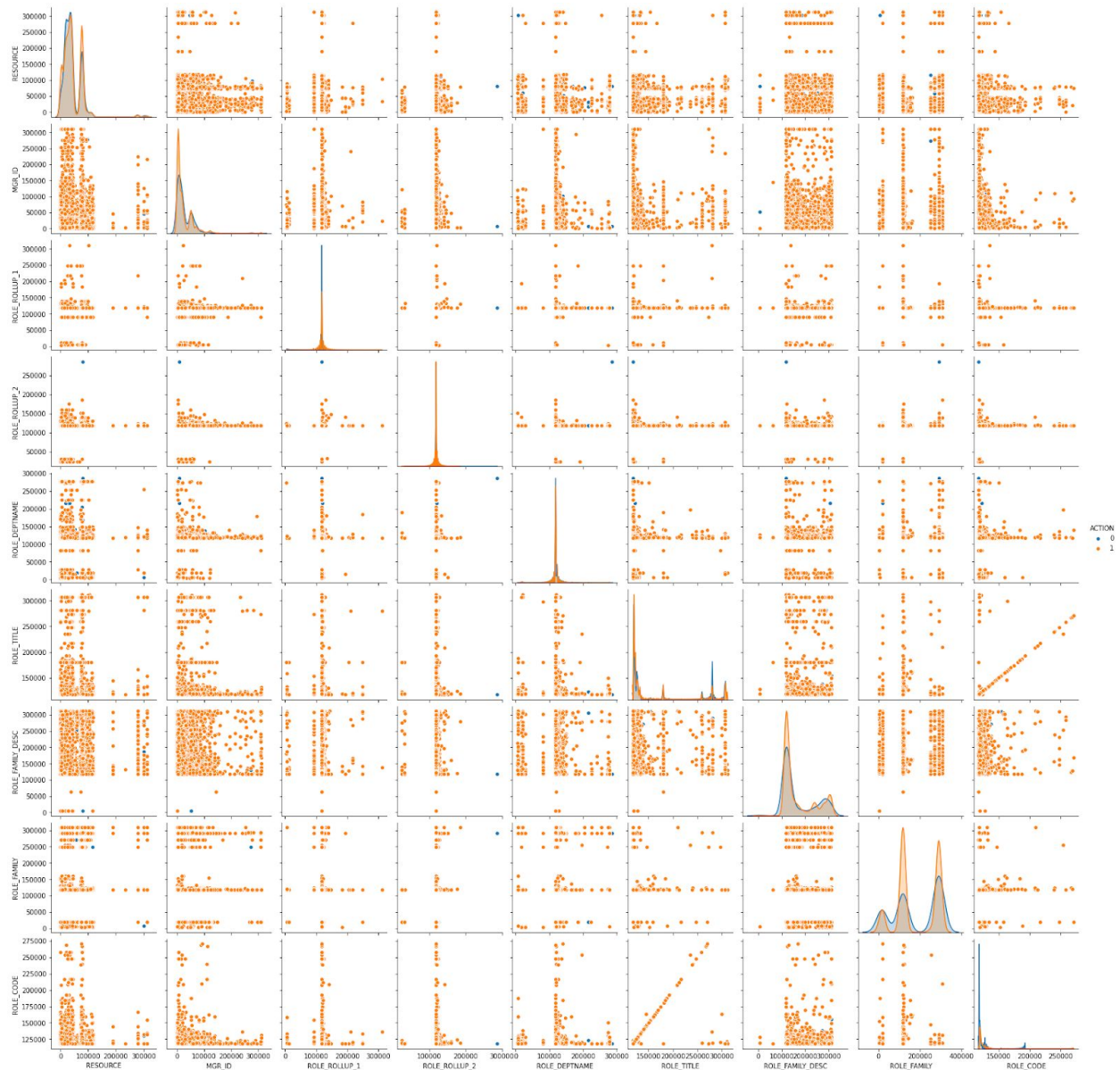
print( len(group_two), len(group_single))

# this mean both of them signifies the same thing and one of them can be removed.

```

343 343

Pair plot for the dataset with ACTION as hue



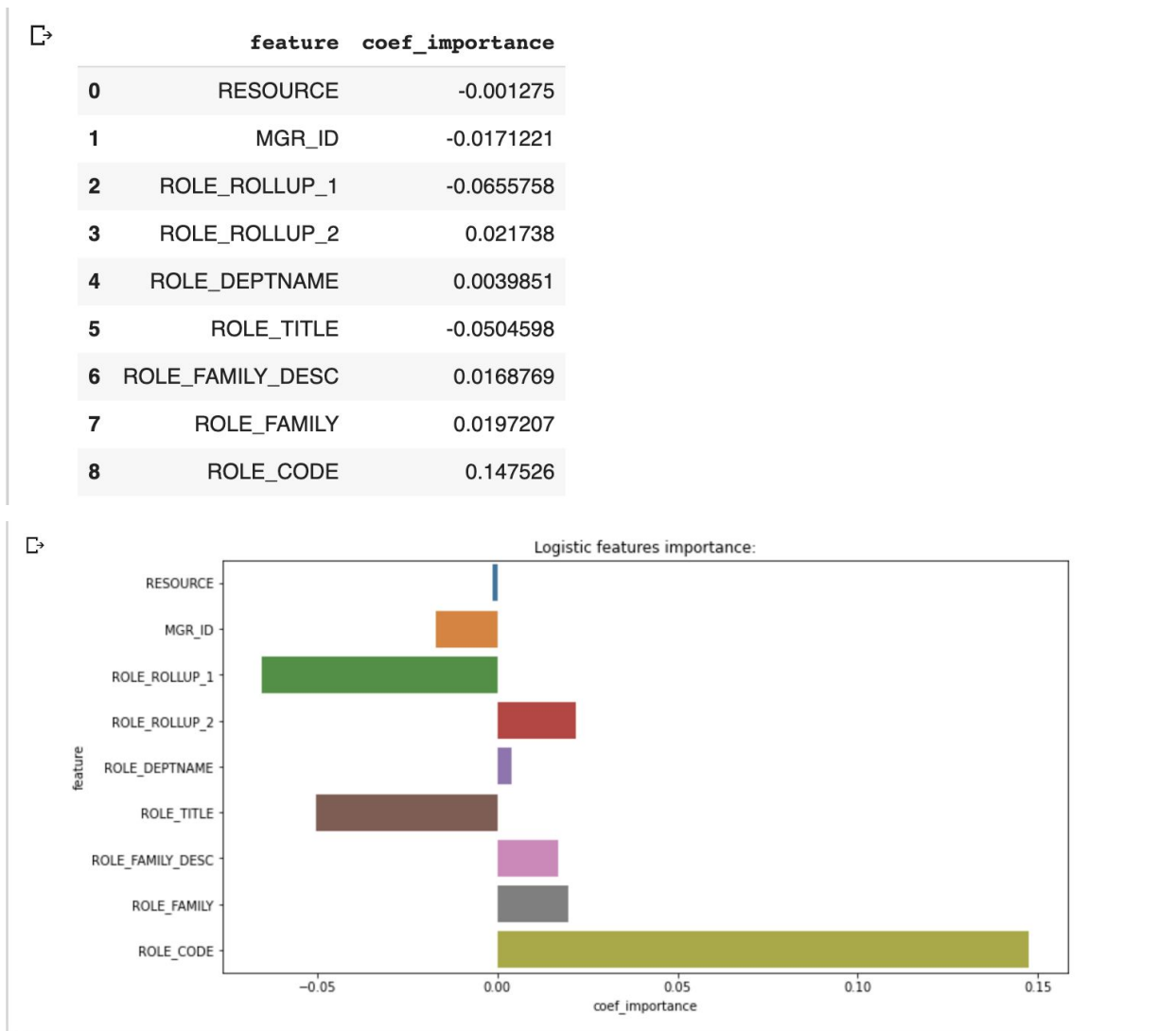
One Hot Encoding

Dataset is encoded using one hot encoding. All the features were categorical in nature. Although algorithms were tried with both original and the encoded dataset.

This resulted in 15283 new columns. This makes the dataset very sparse but more useful for models.

Section 4: Modeling

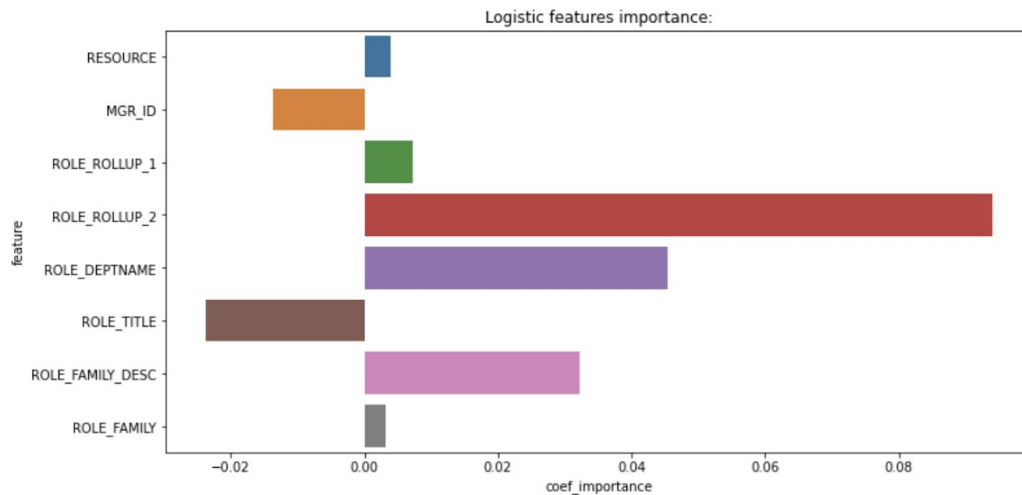
These were the results before removing the column Role_code.



Here first I tried to find the important features with respect to logistic regression, and roll rollup1 and role rollup2 and role code are important features. Then with the Logistic regression, by getting the coefficients and plotting their magnitude, we can see that role code has the magnitude which is highest from the rest.

Here I tried to find important features with respect to the Extra tree classifier model. These are different from the one from the Logistic regression model. Here I got a resource, Manager id and role department, which makes sense as well. As these features determine whether a resource will be given to employees or not.

Removing the Role_code affected the feature selection. It is significantly seen from the Figure ___. As in case of logistic regression, role_code significantly has the highest value of coefficient, therefore removing it increased the value from 0.02 to 0.09 for Role roleup 2.



Role_code included	Algorithms	Features Selected
Yes	Logistic regression	ROLE_ROLLUP_1, ROLE_ROLLUP_2 and ROLE_CODE
Yes	Extra Tree Classifier	RESOURCE, MGR_ID and ROLE_DEPTNAME
No	Logistic regression	ROLE_ROLLUP_2
No	Extra Tree Classifier	RESOURCE, MGR_ID

No.	Model	Evaluation metric	Features	One Hot encoded	Result
1	LogisticRegression	accuracy score	All except Role_Code	No	Test Accuracy 0.94 and Train Accuracy 0.943
2	Extra Tree Classifier	accuracy score	All except Role_Code	No	Test Accuracy 0.942 and Train Accuracy 1
3	Extra Tree Classifier	accuracy score	RESOURCE', 'MGR_ID', 'ROLE_DEPTNAME'	No	Test Accuracy 0.935 and Train Accuracy 0.997
4	CatBoostClassifier	roc_auc with cross validation	All except Role_Code	No	Test roc_auc 0.544 and Train roc_auc 0.55
5	LogisticRegression	roc_auc with cross validation	All except Role_Code	Yes	Test roc_auc 0.8660 and Train roc_auc 0.974
6	CatBoostClassifier	roc_auc	All except Role_Code	Yes	Test roc_auc 0.5 and Train roc_auc 0.5
7	CatBoostClassifier	accuracy score	All except Role_Code	Yes	Test Accuracy 0.94 and Train Accuracy

					0.944
8	Extra Tree Classifier	roc_auc with cross validation and kfold	All except Role_Code	Yes	Test roc_auc 0.736 and Train roc_auc 0.784
9	Extra Tree Classifier	roc_auc with cross validation and kfold	RESOURCE', 'MGR_ID', 'ROLE_DEP TNAME'	Yes	Test roc_auc 0.679 and Train roc_auc 0.748

Mostly in other notebooks, they encoded the variables by defining user defined functions. Thus, I only tried One hot encoding which is inbuilt in the preprocessing library.

Logistic Regression Classifier

In Logistic Regression, we wish to model a dependent variable(Y) in terms of one or more independent variables(X). It is a method for classification. This algorithm is used for the dependent variable that is Categorical. Y is modeled using a function that gives output between 0 and 1 for all values of X.¹

Since the target variable is categorical, it seems a good choice to apply this model on the dataset. Advantages of this models:

It is a widely used technique because it is very efficient, does not require too many computational resources, it's highly interpretable, it doesn't require input features to be scaled, it doesn't require any tuning, it's easy to regularize, and it outputs well-calibrated predicted probabilities.² It can be used as a baseline to compare with other models as well.

Parameters: I used a Liblinear solver. I also tried Saga solver but there was no significant difference in the results and it took longer to run. Also max_iters were 1000. With Stratified K fold of 5 splits and cross validation, an average of 0.866 roc_auc scores comes out with one hot encoded dataset. While with the original dataset, 70:30 split ratio, random state 99, the accuracy comes out to be 0.94 for testing.

¹ "Logistic Regression: A Simplified Approach Using Python." 17 Sep. 2018, <https://towardsdatascience.com/logistic-regression-a-simplified-approach-using-python-c4bc81a87c31>.

² "Real world implementation of Logistic Regression - Towards" <https://towardsdatascience.com/real-world-implementation-of-logistic-regression-5136cefb8125>.

For this problem, the accuracy metric was roc_auc for the original problem. Though the results from the accuracy scores are significantly high and similar for every algorithm. Also the kaggle submission of this model gave a public score of 0.5

Extra Tree Classifier

Extremely Randomized Trees Classifier(Extra Trees Classifier) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a “forest” to output it’s classification result.³ The Extra Trees classifier performed similarly to the Random Forest. Extra trees seem to keep a higher performance in presence of noisy features.⁴ Also the nodes are split based on random splits among a random subset of the features selected at every node. Therefore it kind of tries different trees and provides solutions with extremely randomized trees. Which results is almost close but better results than random forest.

The results were kind of similar for the case of comparing the accuracy scores (0.943). With Stratified K fold of 5 splits and cross validation, an average of 0.736 roc_auc scores comes out with one hot encoded dataset. While with features 'RESOURCE', 'MGR_ID' and 'ROLE_DEPTNAME' an average of 0.6798 roc_auc scores comes out with one hot encoded dataset. Therefore trying with selected features decreased the test score as well the train score. Also the kaggle submission of this model gave a public score of 0.67.

Cat Booster Classifier

CatBoost is a high-performance open source library for gradient boosting on decision trees.

Various advantages of using Cat booster Classifier⁵:

- Performance: CatBoost provides state of the art results and it is competitive with any leading machine learning algorithm on the performance front.
- Handling Categorical features automatically: can use CatBoost without any explicit pre-processing to convert categories into numbers. CatBoost converts categorical values into numbers using various statistics on combinations of categorical features and combinations of categorical and numerical features.
- Robust: It reduces the need for extensive hyper-parameter tuning and lowers the chances of overfitting also which leads to more generalized models. Although, CatBoost has

³ "ML | Extra Tree Classifier for Feature Selection - GeeksforGeeks."

<https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/>.

⁴ "Random forest vs extra trees – The Kernel Trip." 2 Sep. 2018,

<https://www.thekerneltrip.com/statistics/random-forest-vs-extra-tree/>.

⁵ "CatBoost: Machine learning library to handle categorical data" 14 Aug. 2017,

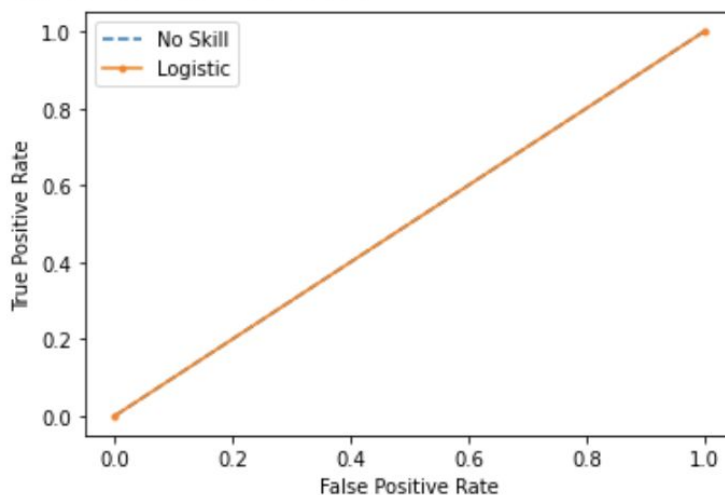
<https://www.analyticsvidhya.com/blog/2017/08/catboost-automated-categorical-data/>.

multiple parameters to tune and it contains parameters like the number of trees, learning rate, regularization, tree depth, fold size, bagging temperature and others.

Since the values were all categorical. The initial thought was that this will produce best results out of all the other algorithms. Though it also gave a public score of 0.5. Even with one hot encoded dataset, results were not significantly high. The roc curve for with respect to the Cat Booster Classifier with both values 0.5 and 0.5.

No Skill: ROC AUC=0.500

Logistic: ROC AUC=0.500



Section 5: Summary and Conclusion

One variable `role_code` was improved as it had one to one relation with `role_title`. The most important features were different according to different models' approaches. They also depended on the other variables as well.

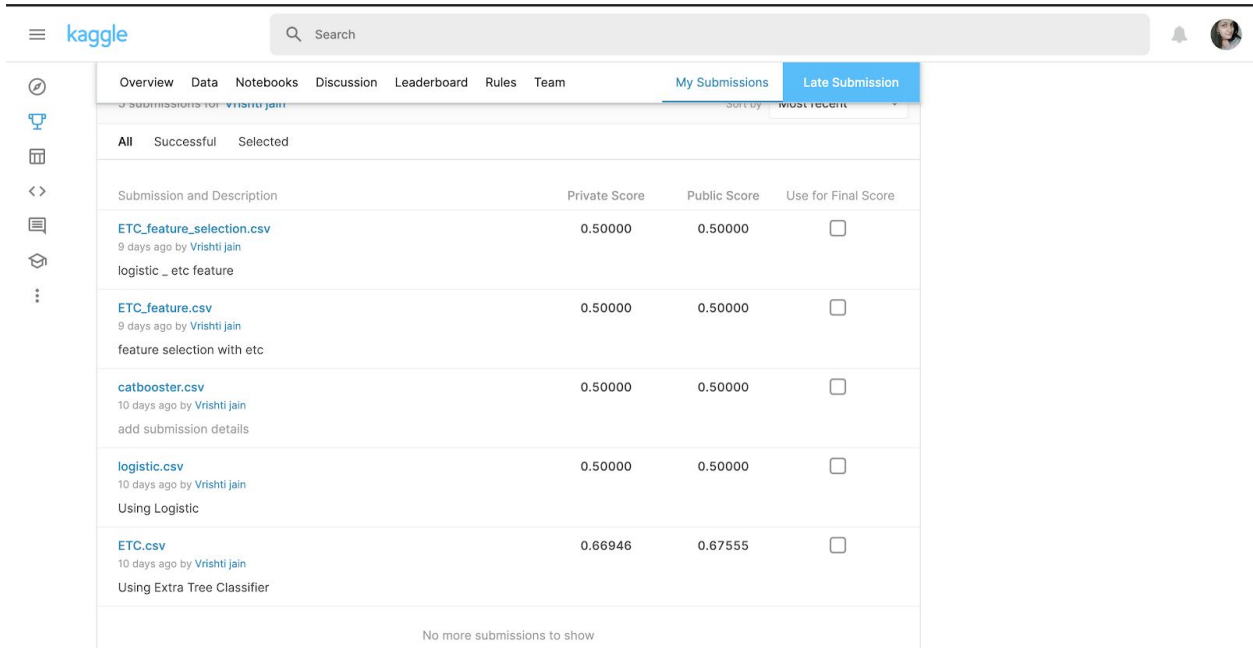
Though with the original dataset, the public score for the Extra Tree classifier turns out to be the best, with one hot encoded dataset, Logistic Regression gave best `roc_auc` score. The Cat Booster algorithm which is supposed to work best with categorical dataset did not work quite with the one hot encoded dataset.

Model	Best Score (roc_auc) on Test data
Logistic Regression	0.8660
Extra Tree Classifier	0.7366
Cat Boost Classifier	0.5

In future, I would like to work more on the encoding techniques and how each of them affects these algorithms.

Appendices:

1) Kaggle Competition score



Submission and Description	Private Score	Public Score	Use for Final Score
ETC_feature_selection.csv 9 days ago by Vrishti Jain logistic _ etc feature	0.50000	0.50000	<input type="checkbox"/>
ETC_feature.csv 9 days ago by Vrishti Jain feature selection with etc	0.50000	0.50000	<input type="checkbox"/>
catbooster.csv 10 days ago by Vrishti Jain add submission details	0.50000	0.50000	<input type="checkbox"/>
logistic.csv 10 days ago by Vrishti Jain Using Logistic	0.50000	0.50000	<input type="checkbox"/>
ETC.csv 10 days ago by Vrishti Jain Using Extra Tree Classifier	0.66946	0.67555	<input type="checkbox"/>

No more submissions to show

2) Link to Kaggle competition:

<https://www.kaggle.com/c/amazon-employee-access-challenge/overview>

3) Link to personal Github repository :

https://github.com/vrishtijain/ML_APPS_Project

4) Link to Project Submission:

<https://github.com/rpi-intro-ml-app-fall-2019/final-project-vrishtijain>