Flight Risk: Defining and Classifying Project Risk for NASA

Submitted to

The National Aeronautics and Space Administration

For

The Risky Space Business Challenge

February 7, 2022

Ainesh Pandey, Sr. Data Scientist at IBM
128 N Craig Street, Apt 710
Pittsburgh, PA 15213
Telephone: (510) 894-5143
E-mail: ainesh93@gmail.com

Table of Contents

1 Introduction

1.1 Abstract

The National Aeronautics and Space Administration (NASA) executes complicated projects prone to a variety of risk. To alleviate these risks, NASA wants to develop an AI/ML solution to categorize and predict risk for future projects. In this white paper, we address this problem by applying topic modeling with LDA to extract risk categories and training a gamut of multi-class modeling algorithms to predict future risk.

Topic modeling with LDA revealed three main categories of risk: technical execution risk, managerial process risk, and operational cost risk. After classifying the risk of each past project, we trained and tuned base multi-class models and developed custom ensembles. Our final model, an ensemble of three base classifiers, performs with a stable 79% accuracy, macro-average F1, and weighted-average F1 with just the projects' title and abstract as inputs. Using this model, NASA can quickly and accurately predict potential pitfalls in future projects and adjust their execution accordingly, increasing future project success rate and leading to fewer adverse outcomes.

1.2 Problem Statement

Through the execution of many complicated and high-cost projects at NASA, several issues develop related to the overall cost, scheduling, technical development, and/or programmatic risk of the project. NASA wants to develop an AI/ML solution that learns from the lessons of past projects to better understand risks associated with future projects.

The solution should:

1. Define the data that needs to be collected and a structure to be used to output risk data for AI/ML use
2. Extract past project data into new output
3. Develop structures for categorizing and identifying risks
4. Design an AI/ML system that uses all past data to identify known and unknown potential risks in new projects

1.3 Background

As an independent agency of the U.S. federal government, NASA is responsible for the civilian space program and space-related research. This charter encourages NASA to take on projects with considerable risk, without which there is no real progress in such an unpredictable discipline. During its 63-year tenure, NASA has learned lessons from the execution of many projects. NASA wants to examine the lessons learned from these projects to evaluate the potential for risk in future projects. Through the Risky Space Business Challenge hosted on the Freelancer platform, NASA hopes to enlist the assistance of external technical practitioners to generate innovative approaches to predict project risks using the lessons learned from past NASA projects.

Challenge entrants are provided several different data sources, including:

- Project reporting files from a couple of NASA's past projects (Astrobee, SynBio)
- Data from the NASA Public Lessons Learned System, as a CSV
- NASA Project Document Summary, including descriptions of all documents provided
- We are encouraged to use external data as well

For the purposes of our analysis, we will focus primarily on the lessons_learned.csv data.

2 Method

2.1 Topic Modeling to Define Risk

The first problem we need to address is defining risk in the context of NASA's past projects. The lessons_learned.csv file represents an English text-based summary of each project's execution, including the following features:

- Identifier Column: *Lesson ID*
- Description Columns: *Title*, *Abstract*
- Organizational Columns: *Organization*, *Project/Program*, *Sensitivity*, *NASA Mission Directorates, etc*
- Lessons/Risk Columns: *Lessons Learned*, *Recommendations, Driving Event, Date Lesson Occurred, Evidence, etc*

The challenge here is extracting some semblance of ground truth regarding the risk incurred during each project, which exists somewhere in the *Lessons Learned* column. The data in this column does not follow any specific format that we can take advantage of, so we need to come up with an innovative way to extract risk categories. To do so, we will apply LDA topic modeling to the *Lessons Learned* column.

2.1.1 An Introduction to Topic Modeling with LDA

Latent Dirichlet Allocation (Li, 2018) is a type of statistical modeling used to classify text in a document to a particular abstract "topic" that occurs in a collection of documents. The algorithm pre-processes the text of each document into a bag-of-words (Brownlee, 2019) or TF-IDF (Stecanella, 2019) representation of each document and builds two models: a topic-per-document model and a words-per-topic model. Together, these models categorize documents into cohesive "topics" that are composed of regularly co-occurring words (or "tokens").

We will apply LDA to the entire corpus (set of *Lessons Learned* values), classifying each document (individual *Lessons Learned*) into topics that we expect to represent risk categories. After pre-processing the *Lessons Learned* column and using industry-accepted methods to identify the ideal number of topics, we can try to better understand the composition of each topic by examining its tokens, determining semantic meaning for each topic (if possible), and extrapolating this semantic definition into a cohesive risk category. We can then classify each project into one of the risk categories extracted.

2.1.2 Pre-Processing Lessons Learned

The raw documents in the *Lessons Learned* column need to be pre-processed into a format that LDA can consume. There are some standard pre-processing steps that we implement, but we need to make some modifications specific to our purposes.

1. We use the **gensim** package's simple_preprocess() function to lowercase, tokenize, and de-accent the text in *Lessons Learned*. This is a standard first step in most NLP projects.
2. We remove stop words (a set of commonly used words in the English language, such as *a*, *an*, *the*, etc.) and keep only tokens longer than three letters. This gets rid of tokens in our text that are likely to offer no value to our analysis.
3. We use the **PorterStemmer** package's stem() and the **WordNetLemmatizer** package's lemmatize() functions to stem and lemmatize each token, which allows us to group together the different inflected forms of a token to consider them as a single token (Rautela, 2021).

After the initial pre-processing, we create a dictionary of tokens using the **gensim** package's **Dictionary()** class. We then filter out tokens that are too common (occur in more than half of the documents) or too rare (occur less than 10 times in the entire corpus) and keep only 10,000 of the most frequent terms to limit the feature space using the **Dictionary()** class' filter_extremes() function.

We now need to convert each document in the *Lessons Learned* column into a TF-IDF representation (Stecanella, 2019). First, we need to perform the intermediary step of converting the documents from the pre-processed token representation into a bag-of-words representation (Brownlee, 2019) using the **Dictionary()** class' doc2bow() function. This bag-of-words representation serves as the input into the **models** package's TfidfModel() function, which produces the TF-IDF representation of each *Lesson Learned* document. Now, the data is prepared for LDA.

2.1.3 Tuning Number of Topics

Identifying the ideal number of topics is a key step of LDA. The metric we use for selecting number of topics is coherence, which scores a single topic by measuring the degree of semantic similarity between high scoring words in the topics. A set of statements or facts is said to be coherent if they support each other (Kapadia, 2019).

The industry-accepted method for selecting the number of topics in LDA is called the "elbow method" (Zvornicanin, 2021). We train the LDA model using a range of number of topics, and the coherence score should generally improve (though not always) as the number of topics increases. This increase will become smaller as the number of topics gets higher. Therefore, we plot the number of topics and their corresponding coherence scores to select the ideal number of topics by visually identifying the elbow in the plot.

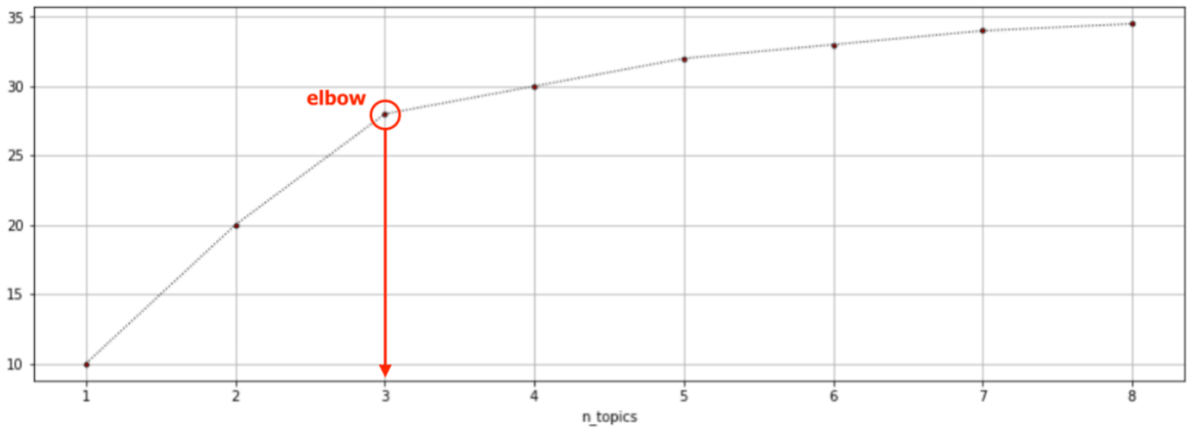

Figure 1: Visualization of Elbow Method to Identify Ideal Number of Topics

2.1.4 Categorizing Projects into Risk Classifications

The LDA model identifies distinct topics and generates words-per-topic functions, with key terms identified for each topic that optimize the overall coherence of the model. Each of the terms has a weight, or coefficient, which identifies its relative importance to that topic. These key terms offer further insight into the overall nature and composition of the topic, and by extension, the risk category they identify.

```
Topic: 0
Words: 0.035*"govern" + 0.024*"open" + 0.018*"coast" + 0.017*"tasmanian" + 0.017*"gold" + 0.014*"australia" + 0.013*"beat" + 0.
010*"win" + 0.010*"ahead" + 0.009*"shark"
Topic: 1
Words: 0.023*"world" + 0.014*"final" + 0.013*"record" + 0.012*"break" + 0.011*"lose" + 0.011*"australian" + 0.011*"leagu" + 0.0
11*"test" + 0.010*"australia" + 0.010*"hill"
Topic: 2
Words: 0.018*"rural" + 0.018*"council" + 0.015*"fund" + 0.014*"plan" + 0.013*"health" + 0.012*"chang" + 0.011*"nation" + 0.010
*"price" + 0.010*"servic" + 0.009*"say"
Topic: 3
Words: 0.025*"elect" + 0.022*"adelaid" + 0.012*"perth" + 0.011*"take" + 0.011*"say" + 0.010*"labor" + 0.010*"turnbul" + 0.009
*"vote" + 0.009*"royal" + 0.009*"time"
```

Figure 2: Example of LDA Topic Functions (Li, 2018)

For example, a topic composed of key terms like "cost", "expens", "money", and "financ" is indicative of a risk category related to cost. Similarly, we examine the key terms of each topic and interpret each topic as a category of risk.

The LDA model then uses these words-per-topic functions to calculate the probability of each document belonging to each topic. The documents are then assigned to the topics with the highest probability calculated. We generate a column in the dataset called *Risk Class* which identifies the topic, or risk category, to which each project has the highest probability of belonging.

2.2 Multi-Class Modeling to Classify Projects by Risk

We now have a semblance of ground truth available to us, extracted from the *Lessons Learned* column by developing an LDA model to classify each project into a category of risk. Now, we need to determine if inputs from the lessons_learned.csv file can accurately predict the category of risk to which a project may belong. With several different risk categories in the Risk Class column, we are now faced with a multi-class modeling task.

2.2.1 Identifying Inputs

The first step of our analysis involves input data selection. The lessons_learned.csv file has the following categories of columns: identifier, description, organizational, and lessons/risk. From this list, we can eliminate the identifier (which has no semantic meaning) and lessons/risk columns (which are related to our output feature and only available post-facto). This leaves the description and organizational columns.

- Description Columns: *Title, Abstract*
  - These columns together capture the purpose of the project. While the *Title* identifies the overarching goal, the *Abstract* delves into further detail.
  - We think that both features together can offer significant value in discerning potential risk.
- Organizational Columns: *Organization*, *Project/Program*, *Sensitivity*, *NASA Mission Directorates, Topics, etc.*
  - These columns contain organizational metadata about the project, such as which group within NASA is responsible for the project, which program it is aligned with, which mission directorates it is tied to, etc.
  - These features may offer further semantic understanding for each project, but we are unsure about the value they bring. It is possible that including these features may introduce noise or information overload to the models we choose to implement.

Based on this understanding of the inputs, we will prototype a solution using just the description columns as inputs. We will then test a solution augmented with organizational columns to see which approach performs better.

2.2.2 Pre-Processing Input Data

We will train supervised multi-class models to predict the risk classification of NASA projects. For this purpose, we need to convert the input data from English text into a tabular format. This requires many of the same standard NLP pre-processing and transformations as topic modeling.

1. We first combine the inputs into one block of text. Depending on the format of the selected inputs, this may require custom processing.
    a. *Organization* has values that may be too short or be eliminated during lemmatization. We need to adjust these values so they do not fall through the gaps of our general pre-processing.
    b. *NASA Mission Directorates*, *Project/Program*, and *Topics* have comma-separated values that need to be combined into cohesive terms.
2. We apply the **gensim** package's simple_preprocess() function.
3. We remove stop words and keep only tokens longer than three letters.
4. We stem and lemmatize each token.
5. We create a dictionary of tokens, extract a bag-of-words representation of each document, and convert it into a TF-IDF representation.

The sparse nature of the resultant dataset may negatively impact the predictive power of many of our modeling algorithms. Therefore, we apply Principal Component Analysis (Volpi, 2020) to convert the inputs from a sparse format into 100 principal components that best capture the variance of the data.

2.2.3 Modeling Approaches

There are several multi-class modeling approaches available, each offering their own unique benefits to the task at hand. Although some approaches generally outperform others, there is no indisputable best option. The only way to identify the best solution is to train and tune multiple classifiers on the same task and evaluate their performance based on the model metrics that are most relevant.

To tune the hyperparameters of each model, we will apply the K-fold cross validation technique. This approach splits the training set into *k* stratified subsets and iteratively trains on *k-1* subsets as the input. Each time, the remaining subset is used to validate the performance of the selected hyperparameters.

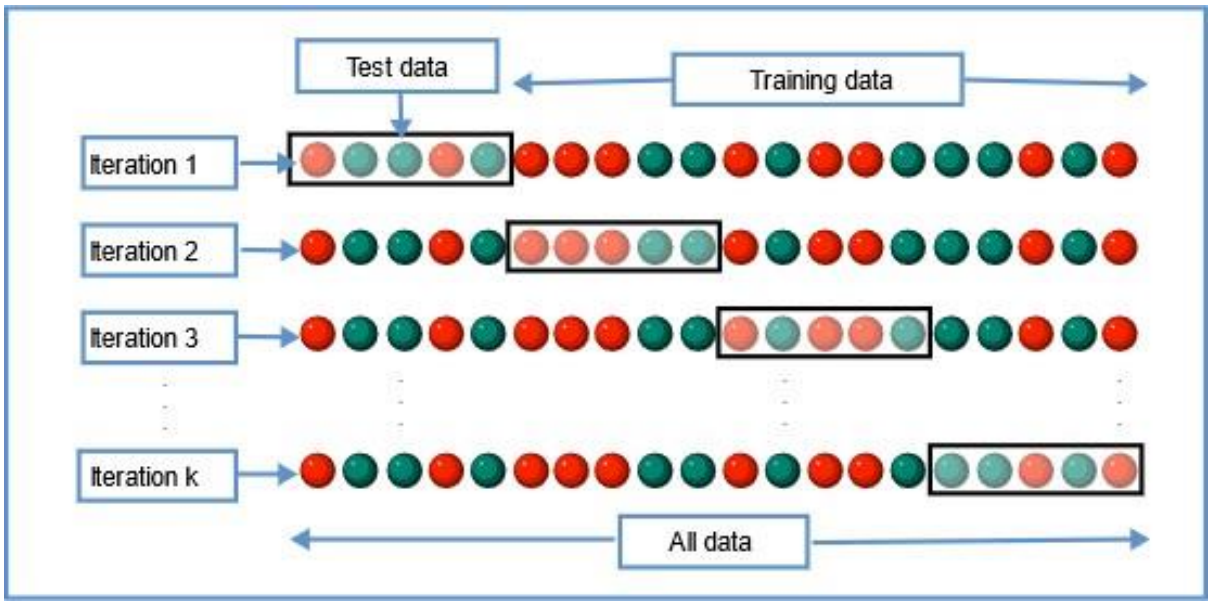

Figure 3: Visualization of K-Fold Cross Validation (V, 2021)

We will be training the following multi-class models in this analysis:

Logistic Regression

- The most basic classifier, logistic regression is easy to implement, interpret, and very efficient to train (Rout, 2020). Although it makes no assumptions about the distributions of classes in the decision space, it has a linear decision surface and therefore does not perform well with non-linear problems. Furthermore, in situations with a low number of records, logistic regression is prone to over-fitting.
- We will tune the following parameters for Logistic Regression:
    - solver: algorithm used in the optimization problem
    - penalty: norm of the penalty
    - C: regularization term

Random Forest Classifier

- Decision trees are non-parametric supervised learning models that learn simple decision rules. An untrimmed decision tree has low bias but suffers from high variance. By ensembling the predictions of many different untrimmed decision trees, random forests lower overall variance and produce more accurate predictions. However, random forests usually require a lot of computational power and training time, and they are not as interpretable as logistic regression models (Random Forest Algorithm in Machine Learning: An Overview, 2020).
- We will tune the following parameters for the Random Forest Classifier:
    - n_estimators: number of trees in the forest
    - max_features: number of features to consider when splitting at each stump
    - max_depth: maximum depth of each tree
    - criterion: function to measure the quality of a split

Light Gradient Boosted Model Classifier

- Instead of learning one complicated classifier, boosted methods learn many weak base learners that are good at different parts of the input space. The output is generated through a weighted vote of each classifier. Light gradient boosted models (LGBMs) use a tree-based learning algorithm that splits the tree leaf-wise instead of level-wise, resulting in trees that grow vertically and not horizontally. Although they are prone to overfitting, LGBMs boast faster training speed, higher efficiency, and better accuracy than most other boosting models (Bakshi, 2018).
- We will tune the following parameters for the LGBM Classifier:
    - max_depth: maximum tree depth for base learners
    - num_leaves: maximum number of leaves for base learners
    - reg_alpha: L1 regularization term on weights
    - reg_lambda: L2 regularization term on weights
    - min_split_gain: minimum loss reduction to make a partition on a leaf node

K-Nearest Neighbors Classifier

- The K-Nearest Neighbors model takes the "birds of a feather" approach to classification. It takes a simple vote of the classification of the K closest data points, using Minkowski distance measures. With no training period, this algorithm is very easy to implement. However, this algorithm does not work well with large datasets, requires scaled or normalized input data, and is sensitive to noise and outliers. Furthermore, models with lower Ks suffer from high variance (Kumar, 2019).
- We will tune the following parameters for the-K Nearest Neighbors Classifier:
    - n_neighbors: the number of neighbors
    - weights: weight function used in the prediction
    - algorithm: algorithm used to compute the nearest neighbors
    - p: power parameter for the Minkowski metric

Gaussian Naïve Bayes Classifier

- The Gaussian Naïve Bayes classifier leverages the Bayes Theorem to estimate the probability of a data point belonging to a certain class. It makes the "naïve" assumption that all features are conditionally independent, given the class label, and it computes probabilities using maximum likelihood estimation. Quick and easy to implement, it can give great results if the conditional independence assumption holds. However, features often show some form of dependency (Naive Bayes Classifier : Advantages and Disadvantages, 2021).
- We will tune the following parameters for the Gaussian Naïve Bayes Classifier:
    - var_smoothing: the portion of the largest variance of all features that is added to variances for calculation stability

2.2.4 Model Evaluation Metrics

To facilitate tuning of each model, we will need to identify the correct model metric to evaluate models against each other. There are several model metrics that we can examine:

- accuracy: simple calculation of correct predictions over total data points
- precision: proportion of data points correctly predicted as a class to the total number of data points predicted as that class
- recall: proportion of data points correctly predicted as a class to the total number of data points belonging to that class
- F1-score: harmonic mean of both precision and recall
- ROC AUC: Area Under Curve of the ROC (receiver operating characteristic) curve used to visualize the trade-off between the true positive rate (TPR) and false positive rate (FPR)
- PR AUC: Area Under Curve of the PR (precision-recall) curve which tells us the average precision (between classes) calculated for each recall threshold

The model metric we select will depend on several critical factors. Firstly, the class balance plays a key role. Some metrics, such as accuracy and ROC AUC, are bad candidates as model metrics for unbalanced datasets because they inherently favor models accurate on the prevalent class, which may not be the goal of our exercise. Secondly, depending on the semantic definitions of each risk class, we may value the accuracy of each prediction (precision) differently than correctly predicting each data point (recall). If both scenarios are equally important, we may need to consider the F1-score. We will select the model metric after understanding more about the risk categories produced in the topic modeling stage.

3 Results and Future Use

3.1 Risk Categorization with LDA

We start by extracting topics from the *Lessons Learned* column of lessons_learned.csv and extrapolating them into cohesive risk categories in the **GenerateRiskTarget.ipynb** notebook.

During the exploration of the *Lessons Learned* column, we identified twenty (20) lessons that were some variations of "See attached report". We removed these rows from further analysis.

After pre-processing and transforming the *Lessons Learned* column from English-text to a TF-IDF representation, we tuned the number of topics by training the LDA models with a range of number of topics from two (2) to fifteen (15). The resulting graph showed that the "elbow" (and in this case, the maximum coherence as well) occurred with three (3) topics.

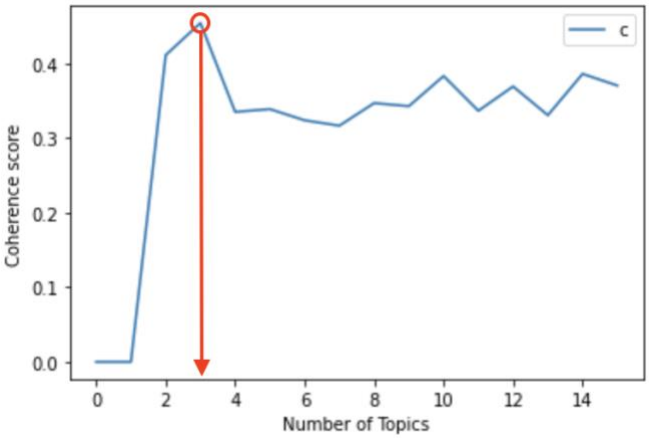

Figure 4: Graph of Coherence Scores vs. Number of Topics

We trained the LDA model to identify 3 topics and generated topics with functions defined by the following key tokens.

Topic 0

Token	Coefficient
failur	0.009
caus	0.009
pressur	0.008
damag	0.008
valv	0.007
instal	0.007
result	0.007
compon	0.007
wire	0.007
load	0.007

Topic 1

Token	Coefficient
project	0.013
requir	0.010
manag	0.010
team	0.009
contractor	0.009
work	0.009
review	0.009
develop	0.008
program	0.008
process	0.007

Topic 2

Token	Coefficient
test	0.016
design	0.010
flight	0.010
hardwar	0.009
spacecraft	0.009
cost	0.008
mission	0.007
risk	0.007
grind	0.007
oper	0.006

We used the LDA model to classify each project into a topic, and we found that the distribution across topics was fairly even: 704 data points in *Topic 0*, 703 data points in *Topic 1*, and 659 data points in *Topic 2*. We now need to evaluate the composition of each topic to determine if they can be extrapolated into risk classifications.

Topic 0

- The key tokens for this topic seem to indicate some technical execution mishap.
- Example of a *Lessons Learned* description with a very high score in this topic: "Excessive pressur or load caus damag to a valv, which result in compon failur."
- This topic can confidently be defined as "***Technical Execution Risk***".

Topic 1

- The key tokens for this topic seem to indicate some issues with the management of the project.
- Example of a *Lessons Learned* description with a very high score in this topic: "Upon review of the work requir to develop the project, the team or contractor did not manag the program well."
- This topic can confidently be defined as "***Managerial Process Risk***".

Topic 2

- The key tokens for this topic seem to indicate planning- or cost-related issues.
- Example of a *Lessons Learned* description with a very high score in this topic: "Inefficient hardware design for the spacecraft led to high oper cost and increased risk for the flight test."
- This topic can confidently be defined as "***Operational Cost Risk***".

These topic classifications can confidently be extrapolated to different categories of risk. We now need to determine if simple metadata about the project, available before the launch of the project, may be effective in predicting the probability of a project belonging to a particular risk class. We save our classifications as risk_classifications.csv.

3.2 Multi-Class Modeling on Risk Classifications

We merge the raw lessons_learned.csv data with the risk_classifications.csv that we developed in the topic modeling phase, and we commence multi-class modeling in the **ModelRiskClass.ipynb** notebook. Because of the elimination of some irrelevant data points during topic modeling, our input space is reduced from 2101 data points to 2066 data points.

3.2.1 Prototyping with *Title* and *Abstract*

We combine the *Title* and *Abstract* into a *Description* column. After pre-processing and transforming the *Description* from English-text to a PCA representation, we split the data into a train and test set. Because we are tuning hyperparameters using cross-validation, we do not need to further define a validation set. We train and tune:

- a Logistic Regression model
- a Random Forest Classifier
- an LGBM Classifier
- a K Nearest Neighbors Classifier
- a Gaussian Naïve Bayes Classifier
- a custom ensemble with all 5 models
- a custom ensemble with the best combination of 4 models
- a custom ensemble with the best combination of 3 models

The **Custom Ensemble: 3 Models** model performs the best on the test set, with an overall accuracy, macro-average F1 score, and weighted-average F1 score of 72%, showing highly stable predictions across all three classes. Now, we augment the inputs to the model to determine if we can improve model performance further.

3.2.2 Augmenting Inputs with Organizational Project Metadata

We analyze the organizational project metadata columns to determine if they should be added as inputs for modeling.

- *Organization* is a straightforward categorical column that provides value and has a manageable number of distinct values. We keep this in our analysis, but we append 'org' to the front of each value so it does not get removed in the pre-processing steps.
- *NASA Mission Directorates* has comma-separated values. There may be value in adding them into our analysis. We reformat the inputs to keep track of each mission directorate separately (e.g. replacing the directorate 'Aeronautics Research' with 'Directorate_Aeronautics_Research' so our analysis will consider it one term).
- *Topics* is formatted like *NASA Mission Directorates*. We handle it similarly (e.g. replacing the topic 'Flight Equipment' with 'Topic_Flight_Equipment' so our analysis will consider it one term).
- *Project/Program* does not have a standard format for its values. Furthermore, it has 200 values for a total of 2000 different projects. We think including this feature in our analysis would add too many columns for not much value. We remove this column from analysis.
- *Sensitivity* has only one value in the entire column. It is useless and is therefore removed from our analysis.

We combine the *Title*, *Abstract*, and selected organizational project metadata columns into a *Description* column and perform the same pre-processing, transforming, and splitting as the prototyping phase. We also train, tune, and test the same models.

The models produced with the augmented data performed worse than the prototype models. The F1 scores (macro-average and weighted-average) and accuracy scores were ~4% lower than the corresponding prototype models. This indicates that the additional features we added to our analysis are adding more noise and are failing to provide additional predictive value for risk classification. This is a positive result because the prototype models are simpler and require much less data preparation.

3.2.3 Best Model Performance Metrics

Our **Custom Ensemble: 3 Models** model is the best-performing model. We can visualize its performance using a confusion matrix, and we can also view comprehensive model metrics using the **sklearn.metrics** package's classification_report() function.

	precision	recall	f1-score	support
0	0.657	0.742	0.697	62
1	0.810	0.797	0.803	64
2	0.695	0.621	0.656	66
accuracy			0.719	192
macro avg	0.721	0.720	0.719	192
weighted avg	0.721	0.719	0.718	192

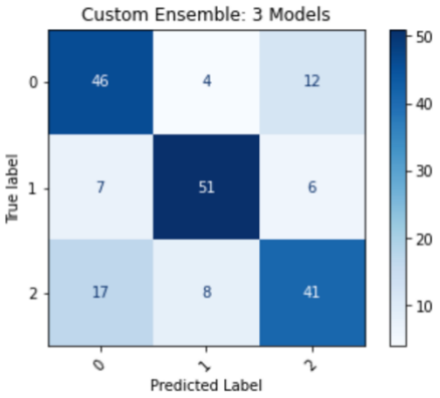

Figure 5: Classification Report and Confusion Matrix for Custom Ensemble Model

3.3 Saving Models for Future Use

This analysis shows that we can effectively use the *Title* and *Abstract* of a project to predict the potential category of risk for future projects. Although the custom ensemble of 3 models performed the best, we save all base models for use by NASA as well. To pre-process the *Title* and *Abstract* into a usable format for the models, the vectorizers used to create the TF-IDF and PCA representations also need to be persisted.

The following files can be found in the **models** directory of the project.

- model_ensemble.pkl
- model_lm.pkl
- model_rfc.pkl
- model_lgb.pkl
- model_knn.pkl
- model_gnb.pkl
- pca_vectorizer.pkl
- tfidf_vectorizer.pkl

Furthermore, the classify_risk.py script has been developed and saved in the **scripts** directory of the project (within the **notebooks** directory) to facilitate two processes for NASA in the future: the batch classification of multiple projects using a CSV file (the **ClassifyRisk_batch.ipynb** notebook serves as an example of this process) or the classification of an individual project using the *Title* and *Abstract* (the **ClassifyRisk.ipynb** notebook serves as an example of this process). Simply update the user inputs (which are heavily documented for ease of use) defined at the beginning of these notebooks to classify future NASA projects.

4 Bibliography

Bakshi, K. (2018, February 25). *LightGBM: A Light Gradient Boosting Machine*. Retrieved from TechLeer: https://www.techleer.com/articles/489-lightgbm-a-light-gradient-boosting-machine/

Brownlee, J. (2019, August 7). *A Gentle Introduction to the Bag-of-Words Model*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

Kapadia, S. (2019, August 19). *Evaluate Topic Models: Latent Dirichlet Allocation (LDA)*. Retrieved from Towards Data Science: https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0

Kumar, N. (2019, February 23). *Advantages and Disadvantages of KNN Algorithm in Machine Learning*. Retrieved from The Professionals Point: http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-knn.html

Li, S. (2018, May 31). *Topic Modeling and Latent Dirichlet Allocation (LDA) in Python*. Retrieved from Towards Data Science: https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24

Naive Bayes Classifier : Advantages and Disadvantages. (2021, July 30). Retrieved from Machine Learning Interviews: https://machinelearninginterview.com/topics/machine-learning/naive-bayes-classifier-advantages-and-disadvantages/

Random Forest Algorithm in Machine Learning: An Overview. (2020, February 19). Retrieved from Great Learning: https://www.mygreatlearning.com/blog/random-forest-algorithm/#AdvantagesandDisadvantagesofRandomForest

Rautela, Y. S. (2021, September 24). *Python | Lemmatization with NLTK*. Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/python-lemmatization-with-nltk/

Rout, A. (2020, September 2). *Advantages and Disadvantages of Logistic Regression*. Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/

Stecanella, B. (2019, May 10). *Understanding TF-ID: A Simple Introduction*. Retrieved from MonkeyLearn: https://monkeylearn.com/blog/what-is-tf-idf/

V, L. G. (2021, May 21). *4 Ways to Evaluate your Machine Learning Model: Cross-Validation Techniques (with Python code)*. Retrieved from Analytics Vidya: https://www.analyticsvidhya.com/blog/2021/05/4-ways-to-evaluate-your-machine-learning-model-cross-validation-techniques-with-python-code/

Volpi, G. F. (2020, May 1). *The most gentle introduction to Principal Component Analysis*. Retrieved from Towards Data Science: https://towardsdatascience.com/the-most-gentle-introduction-to-principal-component-analysis-9ffae371e93b

Zvornicanin, E. (2021, December 7). *When Coherence Score is Good or Bad in Topic Modeling?* Retrieved from Baeldung: https://www.baeldung.com/cs/topic-modeling-coherence-score