Machine Learning for Document Classification

1 Executive Summary

With ever growing amounts of *born-digital* information reaching the age of selection or deletion, there are not enough well-trained eyes available to manually parse and appraise the required documents. The risk of losing records of great historical importance becomes more and more prevalent. The National Archives have therefore identified a critical need to establish machine learning techniques as common practice for record management in many Government departments.

Well trained machine learning algorithms can understand and interpret huge volumes of documents in manageable timeframes, ensuring crucial data is identified and selected for preservation just as accurately as a human archivist would.

However, with the risk of losing vital data, why delete anything at all? The speed of digital adoption has meant that common, unimportant, ephemeral conversations now occur more frequently and via digital communication methods. Without the ability to locate and delete this information, it would clog up digital storage drives indefinitely.

This report will explain how cloud native services hosted in Microsoft Azure can be utilised to resolve the most pressing challenges in the information management space. The implementation method employed by Adatis uses machine learning in innovative ways to assist with the final *preserve* or *delete* decision for a document and allows for enriched content to be fully searchable through a clear, functional user interface.

The results of this research conclude that the architecture deployed by Adatis can locate and classify documents of historical importance in a way that is cost efficient, easy to operate by non-technical users and flexible to a wide spread of document types. In addition, we found that our solution is naturally resilient to problematic content and can be easily extended with custom models.

2 State of the Artificial Intelligence Market in the Information Management / Compliance Space

Artificial intelligence is becoming more prevalent in a larger number of industries and as the technology on offer becomes better and more accessible, this accelerated rate of adoption only increases. In particular, the information management and compliance sector have a lot to gain from these advances, due to the inherent nature of the challenges at hand. Whilst there are several products and services making progress in this area, Microsoft is tackling these challenges head on with a diverse range of practical, innovative, easy to use, tools and services.

2.1 Azure Cognitive Search AI Enrichment Skills

Azure Cognitive Search is an innovative, market leading product from Microsoft that provides rich, intelligent search capabilities over an organisation's entire collection of documents. The service is hosted in Azure and natively can *crack* many different types of file and index their content into a searchable document database. Alongside this capability, Artificial Intelligence (AI) is enabled through a no-code interface to easily cover off common requirements such as entity extraction, key phrase extraction, language identification and sentiment analysis. These AI enrichments can then be stored alongside the searchable text, thereby giving users the ability to, for instance, search for positive content or documents written in French. These AI enrichments that are commonly referred to as skills are being continually added and upgraded with some of the latest built-in, no-code additions including:

- **Text Translation**: This skill translates text at the document cracking stage, this is where documents are initially read, and returns the content into the target language, allowing for a single search language to be used over all documents, regardless of their originating dialect.
- **PII Detection**: This skill can be used to detect and extract a variety of personal information and provide options to mask this data in various ways.
- **Image Analysis**: This skill provides the ability to extract images from documents, understand the subject of the image and store relevant tags in the search index. Additional capabilities include facial recognition and analysis.
- **Custom Entity Lookup**: This new skill allows users to specify a list of words or phrases which will then be fuzzy matched in all incoming documents. Any documents containing any of these custom entities will be flagged.
- **Custom Document Cracking**: This skill allows developers to configure custom steps before the document cracking stage, such as decrypting content or fetching additional data files.
- **Optical Character Recognition**: The skill ensures that text embedded in images is as accessible as normal text.

In addition to these built-in AI skills, custom AI skills can be incorporated using simple REST requests. Data scientists can build a custom model, for example a topic recognition model, and host this in a web service so that the outputs of the model can be stored in the search index alongside the rest of the document metadata. The combination of built-in AI skills and the ability to easily integrate custom skills makes Azure Cognitive Search a truly unique offering to the market that can be incredibly flexible to the needs of the organisation.

2.2 Azure Cognitive Search Ecosystem

With Azure Cognitive Search being the central point for this document search and enrichment capability, several services and tools are positioned around it to maximise its impact in an organisation. The first to mention is the Knowledge Store, which allows users to deposit enriched documents not just in the index but also in an external storage account such as Azure Blob Storage or Azure Table Storage. Documents stored in blob storage can easily be read by any data processing engine such as Spark or Azure SQL Data Warehouse whilst data stored in table storage can be visualised using Power BI. This capability means that Cognitive Search is more than just a deep text search tool but can also facilitate in depth analysis requirements.

Another piece of the ecosystem is the Search App, which is a user interface that can be downloaded from the Azure Portal offering the ability to search, group and filter documents contained within the

index. This tool can be run by any user and requires no-code to be written for it to work although as part of this research we extended its functionality to include buttons that allow the user to update a selection prediction. Additionally, the look and feel of the tool was customised with the use of branding, additional fields, and images to make the tool feel more intuitive to the user.

The search service can also be integrated with Azure Form Recognizer, which means that structured documents can be understood and indexed as if the information was entered directly into the index. The form recognizer can pick up on key-value pairs but also can be customised by a user so that set fields of a form can be labelled according to their contents. This is very useful when form fields are unlabelled and means that routine documents can be interpreted in a very intuitive way.

Finally, the Cognitive Search ecosystem is very secure, with the ability to create private endpoints for secure connections via virtual networks and user managed encryption keys, which can then be stored and manged through key management software such as Azure Key Vault.

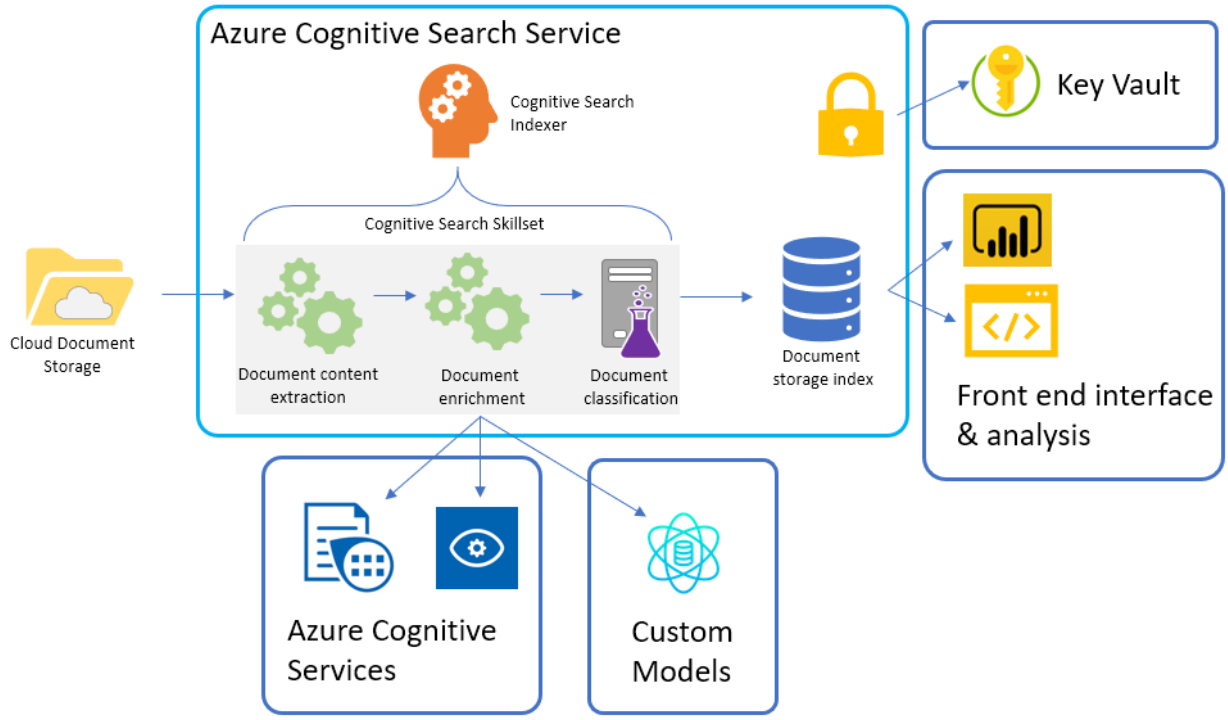


Figure 1: A diagram showing the core Azure Cognitive Search product with the surrounding technologies that make up the eco-system.

2.3 Additional Material

Included in the research are some additional documents that go deeper into the market position of Azure Cognitive Search. The below documents are included in the "Additional Material" folder of the submission pack

- Harvard Business Review – Knowledge Mining
- Extract Actionable Insights from all your Content

3 Service Overview

Azure is the cloud offering from Microsoft that rivals the likes of Amazon Web Services and Google Cloud Platform, containing a wealth of established technologies that allow developers to perform all manner of tasks natively in the cloud. Whilst Azure has well established technologies for nearly all IT disciplines some of its key strengths include data storage, data processing and Artificial Intelligence. Due to the alignment between the challenges of document classification and the strengths of the Azure platform, Microsoft Azure is an excellent choice when looking to tackle this problem.

Unlike an "out-of-the-box" product, a solution built on Azure is an amalgamation of cloud services that, when properly integrated, can be used for many purposes in many industries. In the solution presented by Adatis the following Azure services are used:

- **Azure Blob Storage**: Scalable storage with a low usage cost and easy to integrate API's
- **Azure Cognitive Search**: A search service that uses AI to enrich documents with metadata. Enriched documents are stored in an intelligent index that provides a variety of search options
- **Azure Cognitive Services**: A suite of pre-built, pre-trained models exposed through simple REST API's for performing common AI tasks
- **Azure Databricks**: A Platform as a Service Spark environment for training and evaluating models at scale
- **Azure Python Functions**: Serverless python functions providing a simple method to integrate data pre-processing and model scoring

The below figure shows the high level architecture of the solution.

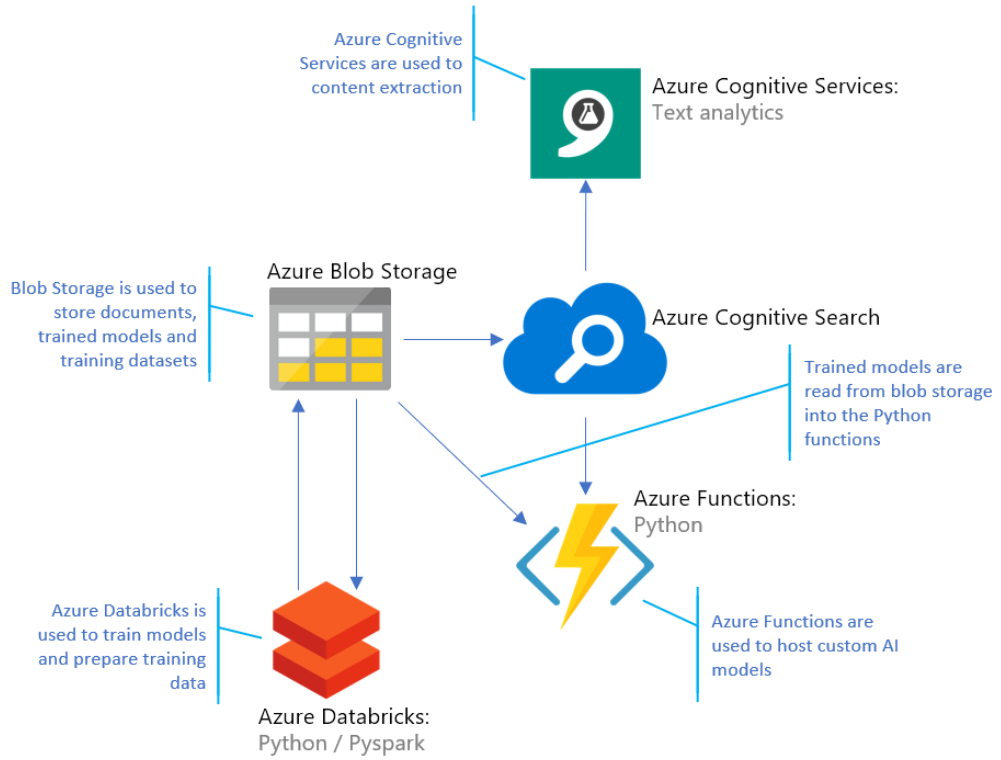


Figure 2: A solution architecture diagram of the Adatis solution

3.1 Azure Cognitive Search

Azure Cognitive Search (ACS) is the engine that underpins the processing and indexation capabilities of the solution by providing the ability to orchestrate document enrichment steps and index each attribute of an incoming document. As a tool for processing documents, ACS has a great benefit in that it can *crack* many different types of common document format and present the content in a single simplified text string and this can include any text from images pulled using Optical Character Recognition (OCR). This saves developers having to write custom code to handle the complexities of reading multiple document and image formats in the same solution. The service also can use out-of-the-box cognitive enrichments such as key phrase extraction and entity extraction as well as custom AI enrichments called using REST requests. In the solution presented by Adatis, the raw content and outputs from out-of-the-box skills are fed into custom skills for further AI enrichment using custom models created and deployed by data scientists.

Once the metadata has been extracted and enriched, the documents are stored in a search index which can make use of complex search algorithms to surface relevant documents to client queries. As well as full text searching, the index can support filtering, faceting (like grouping) and sorting of records by any of the attributes stored in the index. A diagram showing the components of Azure Cognitive Search is below:

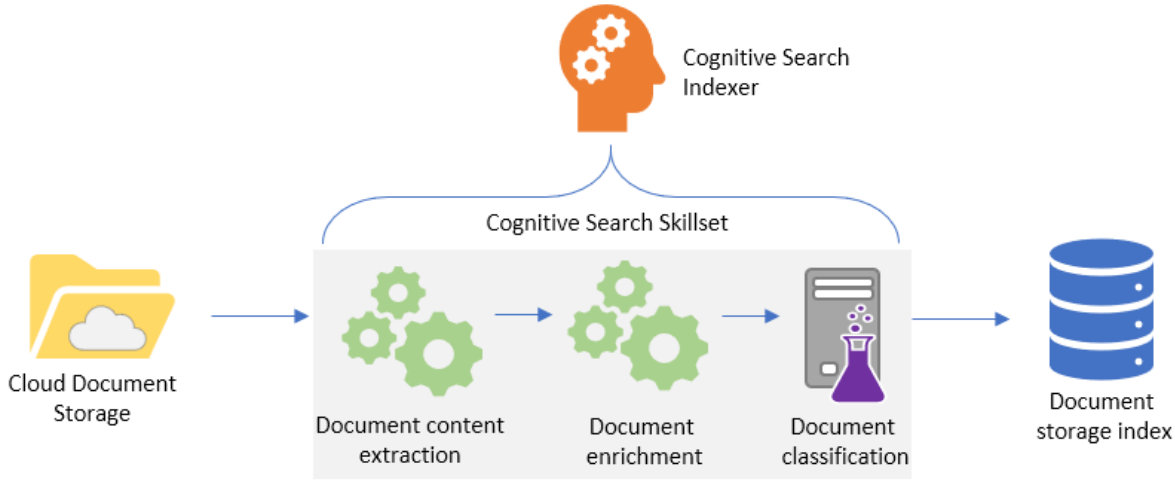


Figure 3: A diagram of Azure Cognitive Search components in logical order.

This link can be used to find our more information on Azure Cognitive Search: https://techcommunity.microsoft.com/t5/azure-ai/knowledge-mining-with-azure-cognitive-search/ba-p/1020774

3.2 Azure Blob Storage

Azure Blob Storage is the most basic storage resource in the Azure platform. It's benefits include a low usage cost and a rich set of API's allowing for many operations to be completed using common tools such as PowerShell and the Azure Portal. The primary reason for its use in this solution is its ability to be integrated with Azure Cognitive Search. By using blob storage, ACS can interrogate the service for a large amount of metadata which can then be used as features in the classification process. Additionally, as with most Azure storage services, blob storage is hugely scalable and can store documents of all types and sizes, offering an assurance that there will never be a document that cannot be stored in Blob storage.

3.3 Azure Cognitive Services

The out-of-the-box services made available through Cognitive Search are powered by Azure Cognitive Services, which are pre-trained AI models exposed via REST calls. Microsoft have a wealth of Cognitive Services for solving all kinds of problems including facial recognition and analysis, anomaly detection and text analytics. Adatis have used these services to automatically extract key phrases and entities from the raw content which can then be scored and understood using custom AI models further down the enrichment pipeline.

3.4 Azure Databricks

Spark has for a long time been the default processing engine when working with large or complex datasets that need to be used to train AI models. Databricks is simply a Platform as a Service implementation of Spark that allows developers to turn off the cluster when it is not in use, thereby saving costs. A feature of databricks is its multi-language support which means that data scientists can work with models in either R, Python, Scala, SQL or soon to be C#.

3.5 Azure Python Functions

For Azure Cognitive Search to integrate with custom AI models, they must be exposed via REST API's and use a common JSON interface for passing and transposing data. In order to implement this kind of interface Adatis have used Azure functions written in python, which is consistent with the rest of the solution but also has powerful capabilities for working with data science models. Further benefits of these functions are that they are very cheap to run, easy to develop and efficient to run and monitor in production.

4 Solution Evaluation

The solution presented by Adatis has functionality that covers off a wide spread of requirements. These include:

- **Data Collection**: Obtaining documents and storing them in a location that is accessible to the tool
- **Pre-processing**: Cleaning and transforming data so that it can be trained and scored as part of the model development or classification pipeline
- **Modelling**: Training and evaluating different classifiers to determine the best algorithm and best set of hyper parameters to give the greatest performance
- **Deployment & user interface**: Deploying the trained model into a running environment where unseen documents can be scored and presented back to users through an easy to use and aesthetically pleasing interface.

4.1 Data Collection

Data can be supplied to the tool through several different routes some of which are considered primary sources, these are Azure data sources natively integrated with Azure Cognitive Search, and others that would be secondary sources, whereby data is ingested into a primary source using an integration tool such as Azure Data Factory. The primary sources include,

- **Azure Blob Storage**: This is storage option favoured for the POC tool developed by Adatis

- **Azure Data Lake Gen 2**: Similar to blob storage with AD security integration and hierarchical name-spacing.
- **Azure SQL Database**: For tabular data this is an excellent option
- **Azure Table Storage**: For larger, less standardised tabular data this would be appropriate
- **Azure Cosmos DB**: Semi structured JSON data would be best stored here

Secondary sources could be data stores in Azure not included in the above list such as Azure Data Lake Gen 1, data stores from other cloud platforms such as AWS and GCP, API end points capable of producing paginated datasets or even on-premises shared drives. Azure Data Factory can run integration pipelines on a regular basis that would connect to any of these storage options and copy the data into the most appropriate primary data source. With this pattern properly implemented it is very unlikely that there is a data source that cannot be integrated with the tool.

4.2 Pre-processing

The tool can read content from several common file formats and can turn this into a simple text string which can then easily be processed further by subsequent transformation steps. The full list of file formats is shown below,

- **Text and office formats**: PDF, docx, doc, docm, ppt, pptx, pptm, xls, xlsx, xlsm, msg (both the message and attachments), plain text files, RTF, EPUB
- **Open Document formats**: ODT, ODS, ODP
- **Structured / Semi-structured formats**: HTML, XML, JSON, CSV
- **Compressed formats**: ZIP, GZ, EML

Amongst each of these formats there may well be images included which can also be processed by the tool and merged into the content string so that image data can be enriched in much the same way as standard text data. Additionally, images can be tagged, described, categorised and even checked for faces. Currently the tool does not support video or audio processing however a custom function could be created that could interface with the Azure video indexer and the response from the video indexer could be stored alongside the other extracted attributes of the document.

Depending on the format that is being read by the indexer, a varying amount of metadata can be returned. If, for example, it is reading from a .msg file then the tool will automatically extract a large amount of information such sender, recipients, cc recipients, bcc recipients, attachments, subject, content, and more. By sampling a several enriched documents it is clear that many document types can expose an author and size metric, with most capable of showing a page count, word count and character count also.

Once the content has been read from the document and loaded into a single text string, enrichment steps are used to gradually extract different pieces of key information about the document before it is ultimately classified. The pre-processing steps used in the tool are listed below,

- **Entity Extraction**: This step uses an Azure Cognitive Service to extract entities discovered in the text and a confidence threshold of 90% was applied to ensure accuracy of results. Entities of interest are locations, organisations and people.
- **Key Phrase Extraction**: This step also uses an Azure Cognitive Service to extract key phrases found within the text which could be poly-grammatic.

- **Data Cleaning**: This is a custom step which uses an Azure Function written in python to clean the raw content, the entity lists and the key phrase list. Cleaning rules can be heavily customised based on the content although in this implementation the function uses Regex to remove any invalid terms and will remove words under 3 characters long. The results of this function are passed back into the indexer for use later in the run.
- **Entity Scoring**: This is another custom step that matches the entities extracted in the first step to a pre-built index of entities that indicate content likely to be selected for preservation. The matching is performed using fuzzy logic to allow tolerances in the lookups and the TF-IDF importance scores for each matched entity are totalled before being returned to the indexer.
- **Key Phrase Scoring**: This custom step is similar to the one previous, as it uses fuzzy matching to match extracted key phrases to an index of important key phrases with the score being totalled.
- **Topic scoring**: This custom step is pre-loaded with a topic model which is trained using Latent Dirichlet Allocation (LDA) and is used to decipher the topics which are hidden in the document content. Once these topics are extracted the confidence scores for each topic are used as features in the classification model.
- **Attribute shaping**: This step uses functionality built into the indexer and allows the extracted features from the previous custom steps to be shuffled into a single json object which can easily be passed into the scoring function
- **Document scoring**: This final custom step uses the features extracted previously to classify documents for selection using an ensemble of 3 classification models.

A diagram of the processing steps described above is shown below:

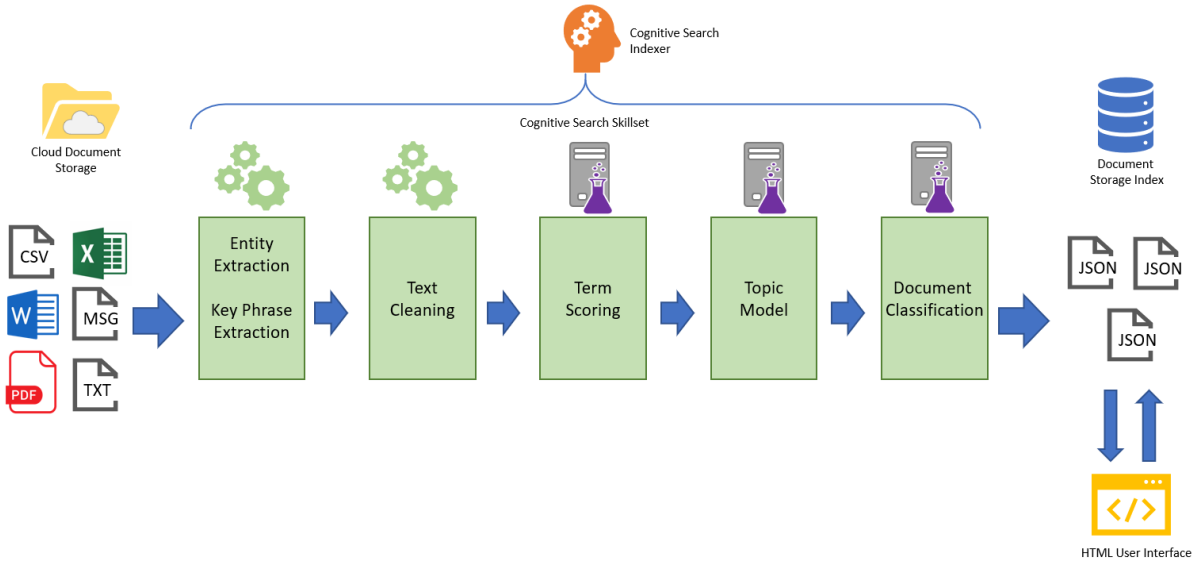


Figure 4: A diagram detailing the processing steps performed as part of the document classification pipeline

The modelling work completed as part of this tool is described in more detail in the next section.

4.3 Modelling

Much of the POC development time was focused around feature generation, model building and refinement. The next section describes the feature engineering process and the types of models that were employed throughout this POC project.

4.3.1 Feature Selection

Feature selection is the process of removing and refining features because they are unimportant, redundant, or outright counterproductive to the training of the chosen algorithm. Sometimes we simply have too many features and we need fewer. An example of a counterproductive feature is an identification variable which doesn't help in generalisation. There are a handful or reasons to remove and refine features including:

- **Focus Training:** Less redundant data means less opportunity to make decisions based on noise.
- **Improves Accuracy**: Less misleading data means modelling accuracy improves.
- **Reduces Training Time:** fewer data points reduce algorithm complexity and algorithms train faster

While manual elimination of statistically irrelevant features is a good initial step in some cases it can also introduce human biases into the analysis and so focus is placed on the importance scores and the correlation matrix to indicate which features to remove. It is also important to consider the feature selection process as part of the model selection process. If the features are first selected and then the model is trained it may introduce human biases, where our human interpretation of the data hinders the statistical interpretation by the model. Also selecting feature prior to training can result in overfitting, where the model is too closely aligned to the training examples and cannot generalise to new data.

To mitigate these issues, feature importance scores are used to evaluate the most important features to the model. The below image shows an initial feature importance plot where the higher scores belong to the most important features for a random forest classifier. This image shows there is some noise in the set of features.

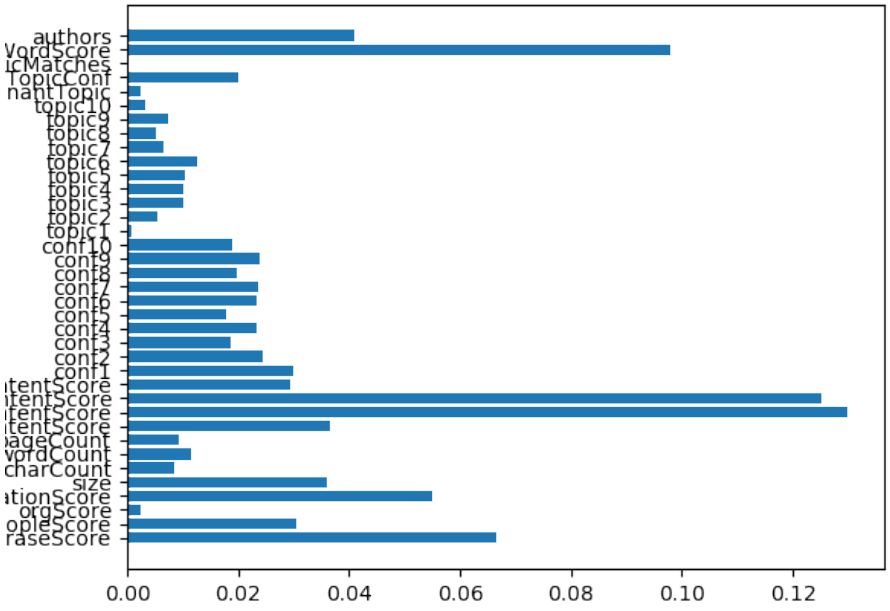


Figure 5: A feature importance plot showing the key features for a random forest classification algorithm.

Based on the plot above, further work was done to reduce the unimportant values and improve the quality of the classification. The updated plot is shown below.

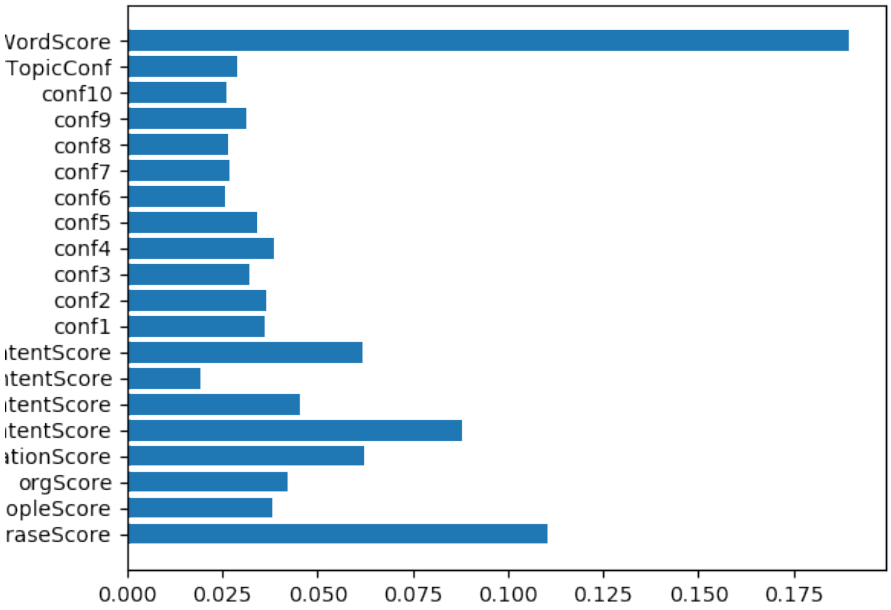


Figure 6: A plot showing a refined set of features based on some preliminary analysis.

Further analysis on features was carried out in the form of correlation mapping. By plotting the correlation of each feature to every other feature, we can identify which features are most related and therefore do not help the model generalise. After an initial round of correlation mapping, a reduced feature set was produced. The correlation map for this new feature set is shown below.

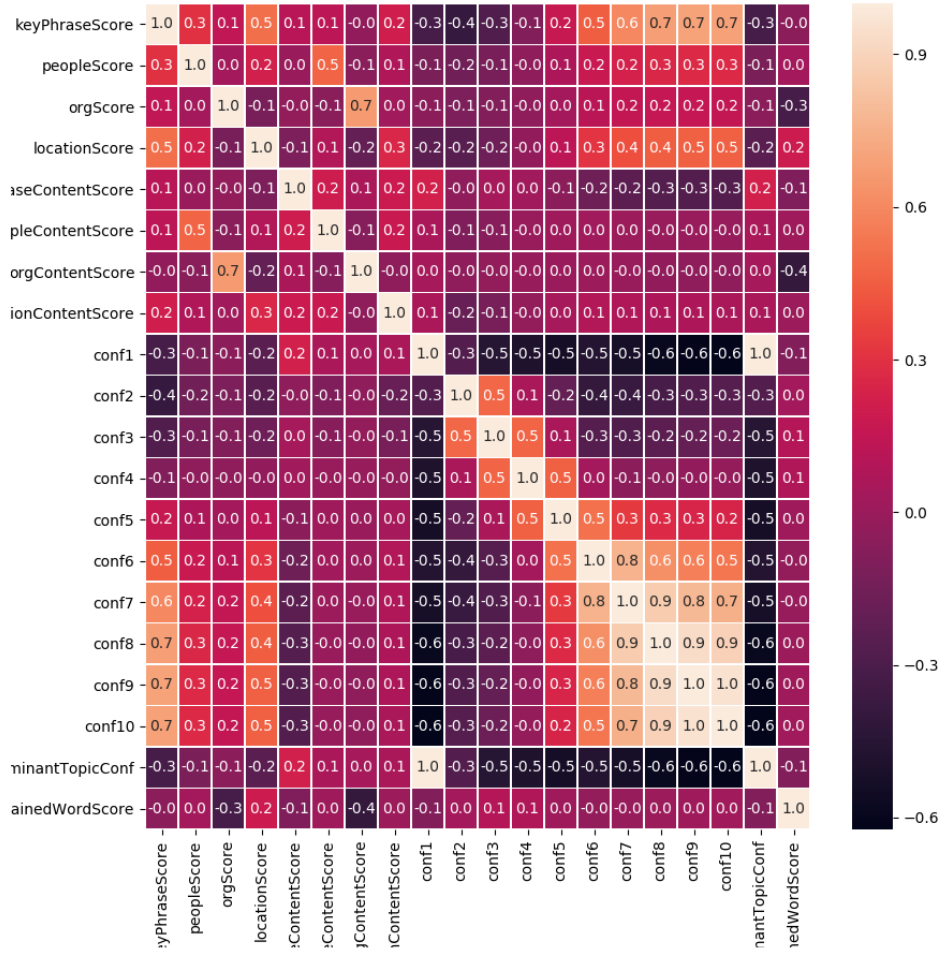


Figure 7: A feature correlation heat map showing the relationship between each feature.

4.3.2 Term Frequency – Inverse Document Frequency

Term frequency – inverse document frequency (TF-IDF) is a method used to score the importance of a term across a group of documents. The algorithm first stores the number of times a term appears in a single document and then divides that number by the number of times the term appears in all other documents in the set. Therefore, the lower the TF-IDF score the more important the term. To make these scores more useful, all preserve labelled documents were processed using TF-IDF and then the extreme values were filtered to remove terms that were too specific or too generic.

As new, unseen documents arrive, the extracted key terms and entities are fuzzy matched against the TF-IDF index to determine how many important terms are found within the document. These scores are then min-max normalized and used as features for the document classification model.

4.3.3 Latent Dirichlet Allocation Topic Modelling

Topic modelling is a branch of unsupervised natural language processing which is used to represent a text document by the inherent topics within itself. The assumption is that the topics extracted from the raw content of a document can best explain the underlying information in that document. The Latent Dirichlet Allocation (LDA) algorithm can find the hidden thematic structure in each document and cluster it into coherent topics. Therefore, each topic is a combination of keywords and each keyword contributes a certain weightage to the topic. The output of the LDA model is a distribution of topics for each document and each topic is a distribution of words.

To train the LDA model a number of pre-processing steps were used to ensure maximum accuracy from the model. The first step is to tokenize the entire document text by splitting the content into single word *tokens* which are then stemmed and lemmatized, removing pluralisation, tense and any other forms of word variation. The tokenised stemmed words are then filtered using a list of common *stopwords* which helps to remove noise from document.

When evaluating the performance of the topic model the following metrics were used,

- **Perplexity score**: This metric captures how *surprised* a model is of new data and is measured using the normalised log-likelihood of a held-out test set.
- **Topic Coherence**: This metric measures the semantic similarity between topics and is aimed at improving interpretability by reducing topics that are inferred by pure statistical inference.

These metrics can be plotted as a line on a graph to reveal the *elbow* point; the point at which perplexity and coherence plateau and any further optimisation no longer increases accuracy. Additionally, the topics and topic terms were visualised as shown below to help assess how interpretable the topic model is.

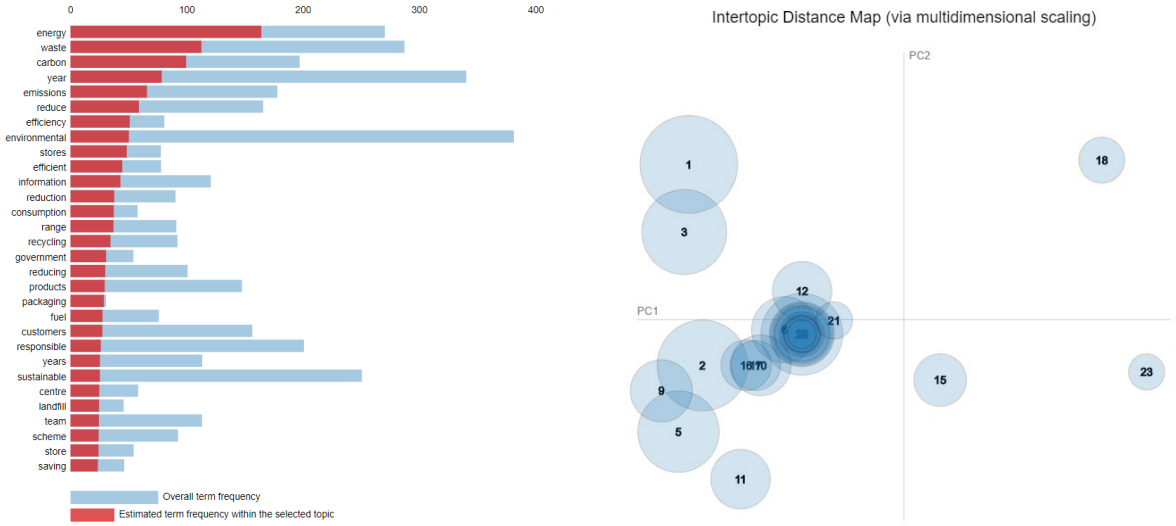


Figure 8 & 9: A plot of the top terms associated to a specific topic and a plot of intertopic distance.

4.3.4 Classification Model Selection

Throughout the research, three classification algorithms were used to provide a better view of how well a classification could be performed. By taking this ensemble approach we achieve a greater degree of confidence in a classification by taking the prediction of the quorum and not just of a single classifier. Whilst many algorithms were initially tested the three that were carried forward through the research were,

- **Random Forest**: A very common model that is used to solve many classification problems. The Random Forest algorithm actually trains a fixed number of decision trees on samples of the entire training set and uses the averaging to improve the predictive accuracy and control over-fitting.
- **Logistic Regression (Binomial)**: This method is very popular for solving binary classification problems and is a better performing version of linear regression, as it uses the natural logarithm function to find the relationship between variables as opposed to a straight line.
- **K Nearest Neighbours**: Another common classification model that uses the labels of the nearest matching points in the training data to assign a label to a new, unknown point. Traditionally, a distance metric is used to determine the nearest neighbours however there is also a dependency on finding the best value for k (k being the number of matches to evaluate).

4.3.5 Classification Model Evaluation

Evaluation and tuning of the selected classification models is an extremely important process, as it is this evaluation that ensures the correct parameters are used and that accuracy is maximised. The primary artefact when reviewing model performance is the confusion matrix as this allows the data scientist to understand the ratios between the correct and incorrect predictions of true and false values. An example of a plotted confusion matrix for the Random Forest model is shown in the below figure.

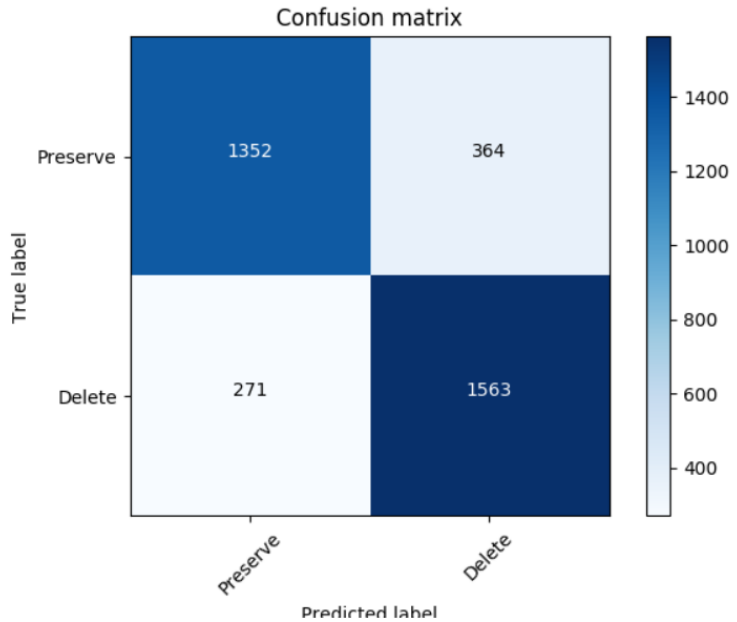


Figure 10: A confusion matrix showing the numbers of true positives, false positives, true negatives and false negatives.

An approach used throughout this research was K fold cross validation which shuffles the dataset to create a set number (k) of test sets, each of these test sets are then passed through the classifier to determine the classifiers predictive ability. Each of the three classifiers were validated using cross validation and an example of the Random Forest result set is shown below. By using the confusion matric produced for the validation fold the following metrics are calculated,

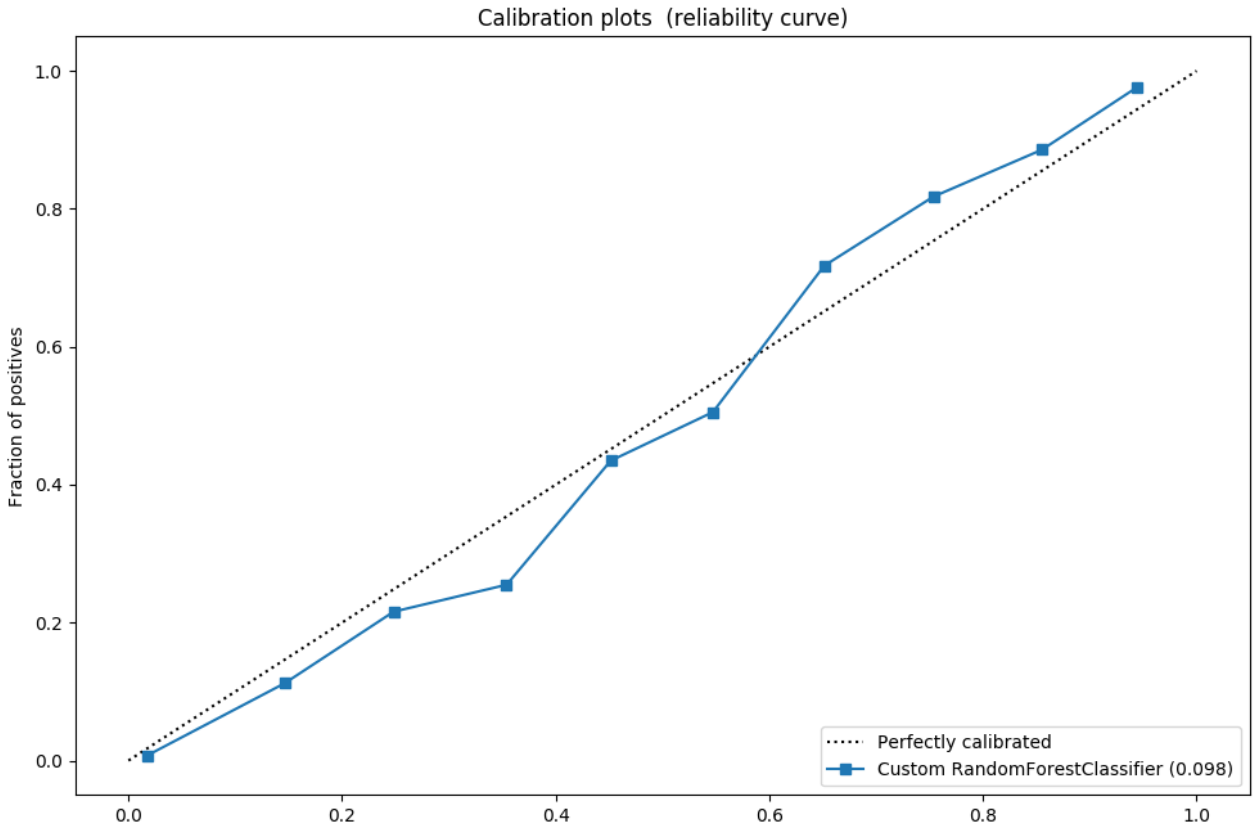
- **Accuracy**: The percentage of correct predictions made by the model for both Delete and Preserve
- **Precision**: The percentage of correctly predicted Preserve labels from cases that are predicted positive
- **Recall**: The percentage of correctly predicted Preserve labels from cases that are actually positive. This is likely the most important metric for this task as it describes the number of documents for selection that were NOT selected.
- **F1 Score**: The harmonic mean of precision and recall allowing for a better optimisation of both metrics.

The below table shows an example of the Random Forest k fold cross validation result set.

estimator	test_accuracy	test_precision	test_recall	test_f1
RandomForest	0.853873239	0.837801609	0.878425861	0.857632933
RandomForest	0.866549296	0.848930481	0.892480675	0.870161014
RandomForest	0.859105319	0.844444444	0.881236824	0.862448418
RandomForest	0.874251497	0.84882199	0.911454673	0.87902406
RandomForest	0.863332159	0.842035691	0.895291637	0.867847411
Avg	**0.863422302**	**0.844406843**	**0.891777934**	**0.867422767**

The full set of k fold cross validation outputs are available in appendix 7.2

To further improve on the predictive ability of the classifiers, a calibration exercise was carried out with the goal of achieving the best (lowest) Brier score. Instead of evaluating a model against simply the prediction, the Brier score uses the predicted probabilities at varying fractions of positive examples to identify if the model is over-confident or under-confident. A perfectly calibrated model will follow the dotted diagonal line closely, whilst an over-confident model will drop below the line and an under-confident model will stray above the line. An example for Random Forest is included in the image below however all three calibration plots are available in the appendices.



4.4 Deployment and User Interface

All the infrastructure used for the tool is deployed into the Microsoft Azure cloud platform and is easily setup and configured. Each resource is named using a consistent naming convention and located within a single resource group to establish good levels of organisation and ensure security boundaries are maintained. An important consideration is the region in which resources are deployed and Azure itself has data centres across the globe. Each resource in this tool is in the West Europe region which ensures that data egress charges are not incurred, and that network latency is minimised. It should be noted that the entire set of resources could be deployed to any of the Azure regions across the globe to ensure the data is stored in closest to the operators without incurring unnecessary costs.

The user interface that was built to accompany the document index is simple by design however performs the required tasks reliably. At this POC stage the UI is an HTML file that has a connection to the document index and can return documents based on search criteria and filters. To assist the record manager with reviewing the classified documents, several attributes have been enabled for faceting. This ability allows documents with the same value to be easily viewed and operated on together, this is particularly useful when updating the selection prediction for a large set of

documents. A further feature of the UI is the ability to update the selection prediction for any documents that are not confidently classified one way or another. This is performed using a simple UI button and performs the update by sending a POST request to the index. This mechanism is simple, reliable and ensures multiple users could be working against the index at the same time as the index supports transactions. The below image shows a screenshot of the UI with some highlighted elements.

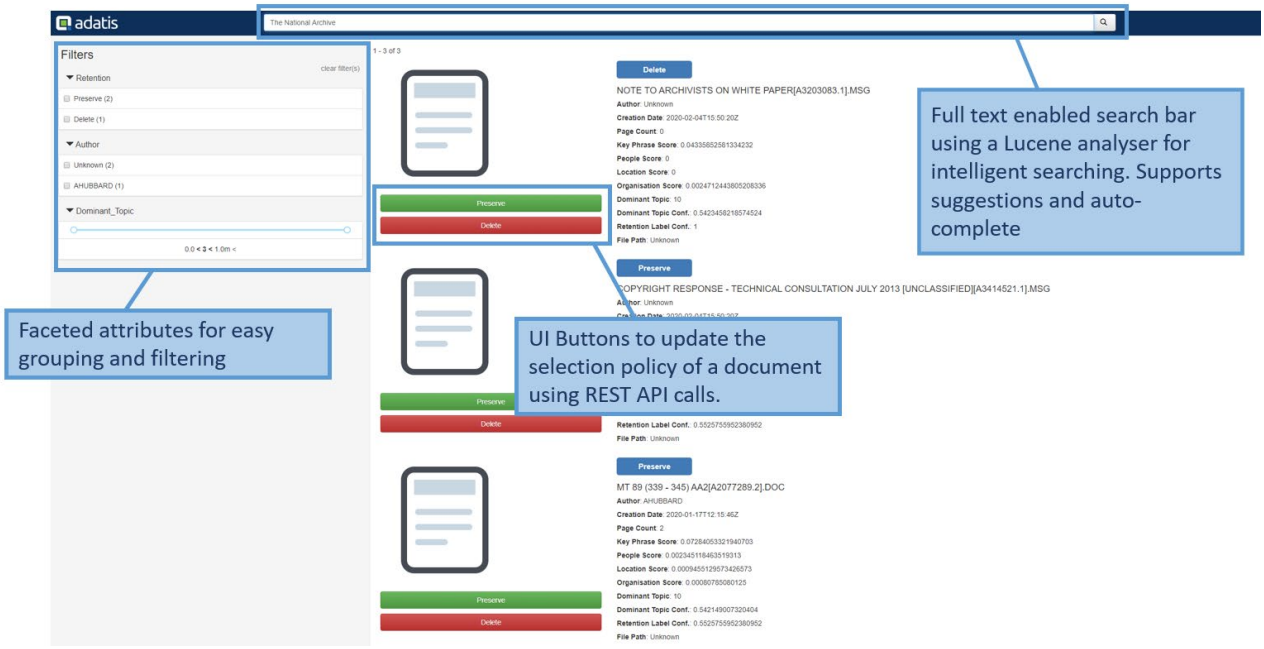


Figure 11: A screenshot of the user interface

The security model for the tool works at two levels. The lower level controls access to the individual resources within Azure and ensures that developers cannot access or delete critical pieces of architecture maliciously or accidentally. This extends to the containers in which documents are stored, thereby ensuring that only service accounts have access to the documents that need to be classified. With this configuration, sensitive documents can be accessed by the tool but not by developers. The higher level is focussed around the user interface and limits the actions that a record manager can perform against a document. This layer of security is not implemented as part of the POC as this was not deemed critical to the minimum viable product.

The levels of skill required to perform various tasks using this tool are varied, ranging from non-technical to data analyst / scientist level, depending on the actions the user is required to complete. Key tasks and a description of the detail required to complete them are listed below,

- **Create a training dataset**: This is a very simple task that could be completed by a non-technical user. The only requirement is that the data set contains the path to each file and a label or similar mechanism to infer whether the document was selected for preservation or not. Given this information the tool will extract the features and train a model using those features however the user is not required to participate in that process.
- **Train a model**: The tool, in its current state, uses Databricks to train models and this means the user would need python knowledge and an appreciation for the model libraries that could be used, however none of this process is proprietary to Adatis. With some further effort the

Azure Machine Learning Designer could be used which provides a *no-code* way of collecting data, training models and deploying reusable processes. This could therefore be carried out by a moderately technical user with no coding or machine learning experience.

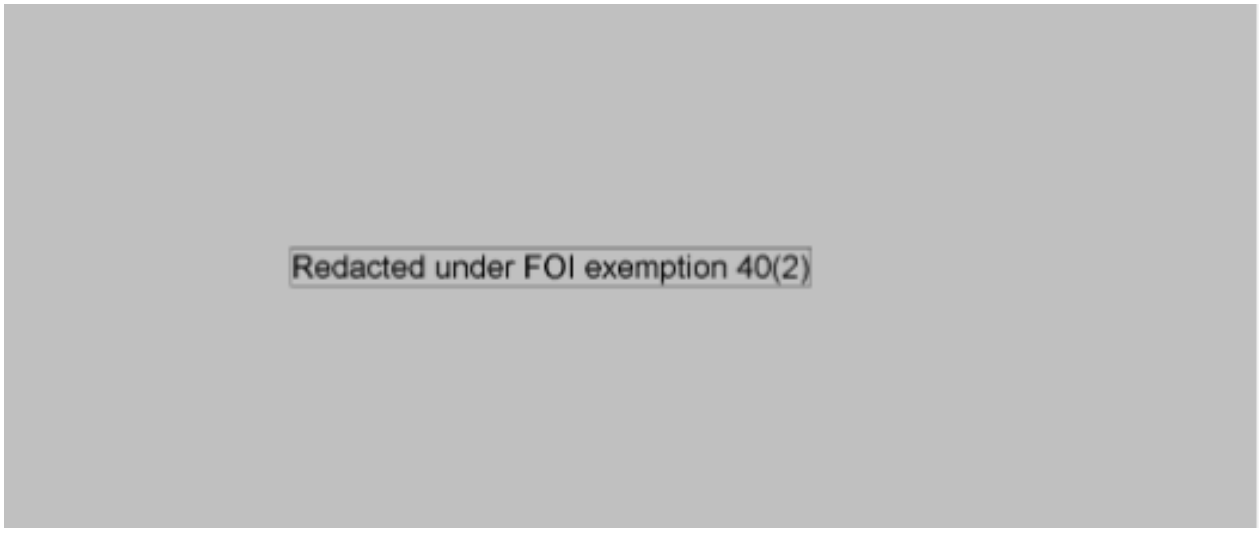
- **Run / score a model**: The tool has the capability to track changes and so the process of scoring a new document is very simple. The user would only have to store the document in the correct storage area and run the indexer process, which could be exposed as a button press in the UI.

5 Results and Findings

5.1 Results

To determine the following results each of the three candidate models was trained on the labelled dataset and calibrated using the evaluation techniques described above. The outputs of this training exercise revealed the best single model was Random Forest, scoring an average recall of 89% and an average accuracy of 86%.

The second part of the results relate to a classification exercise performed over 4,500 randomly sampled documents from the unlabelled dataset with the goal of surfacing any documents that should be selected for preservation. By classifying each document with all 3 models and only selecting ones where all 3 agreed we gained a higher average accuracy overall. The results of this exercise are summarised by the top 10 documents, sorted by the probability of a preserve prediction. Each of these top 10 documents are briefly described in the table below and highlight the models' ability to detect documents of potential historical importance. The full documents are included in the accompanying "Top 10 Preserved Docs" folder.



5.2 Findings

Throughout the course of this POC project several key findings were discovered that are documented below.

5.2.1 The classification model generalised well

At the start of the project a data profiling activity discovered that the documents contained in the unlabelled data had some differences compared to the data that was labelled and used for training. These are described below:

- **Document Type**: The labelled data was comprised of many .msg files whereas the unlabelled data was more mixed between PDF's, Docs and txt files. This difference could be important due to the nature of content in emails vs more formal documents
- **Document Size**: Similarly, the labelled data had a bias toward smaller files whilst the unlabelled data contained some much larger documents.
- **Document Content**: Given the operational purpose of the labelled data, there was a concern that the overall content would not overlap with the unlabelled data, as the unlabelled data is primarily made up of website archives.

As a result, there was a concern that the classification model would not generalise well, proving to be less accurate when classifying new types of document. However, by reviewing the evaluation results based on unlabelled data we found that the model generalised well against this unseen data, meaning that a classification model trained with the specified features can be effective when exposed only to subset of document types.

5.2.2 The topic model was over-saturated

After some initial classification results were derived, the importance and quality of each feature was analysed to assess its need for improvement. A major finding at this stage was that the topic model was over saturated with terms and this meant that documents that should not be selected were scoring too highly from the topic model. Due to the nature of the LDA model it was challenging to address this issue, as the model would see each term equally. However, if this task were to be attempted again the topic model could be made more useful by curating the documents that would be used to train it.

5.2.3 The tool was cost efficient

Each aspect of the tool has a cost associated to it and as the tools are deployed on Azure the cost of each resource can be optimised to ensure the user only pays for what is truly used. As such, the tool is cost efficient and ensures each implementation of the tool can be tailored to the specific usage. If the user has a large number of documents with complex processing requirements, you would expect the cost to be a high however a user with much smaller data volumes will benefit form a greatly reduced cost, and not be tied to a one-size-fits-all style pricing model.

A secondary element to this finding is that the Azure Cognitive Services, key phrase extraction and entity extraction, had a larger cost then all other components. The evidence gathered throughout

this research suggests that £500 would cover the Azure consumption cost for key phrase and entity extraction of roughly 25,000 documents.

5.2.4 Resilience is built into the tool

An early concern about the tool was how it would handle large volumes of documents. This is because there is a known 2 hour limit on indexer runs that use AI enrichments, as this one does, and there is a limited ability to ensure only valid content is presented to the indexer. Fortunately, the first part of the concern is dealt with automatically as the indexer will store the last good document id when it reaches its limit. This means that when the indexer is restarted, either by a schedule, manual invocation or REST API call it will continue from this point. This last good document id can be reset manually if the indexer needs to re-process the entire set of documents again.

The second part of this concern is resolved with a small, but manual, update. By adding the following configuration parameters to the indexer, we could ensure it would remain fault tolerant regardless of invalid documents or content.

```
"parameters": {
    "batchSize": null,
    "maxFailedItems": -1,
    "maxFailedItemsPerBatch": -1,
    "configuration": {
      "dataToExtract": "contentAndMetadata",
      "parsingMode": "default",
      "failOnUnsupportedContentType": false,
      "failOnUnprocessableDocument": false
    }
  }
```

6 Future Research and Development

The development completed during this research phase delivered on a number of critical pieces of functionality however there are a number of goals that could be achieved were there to be more time.

6.1.1 Incremental Indexing

Incremental indexing is currently a preview feature of Azure Cognitive Search, meaning it is available for use but should not form a major part of a production system. This feature allows the system to cache previously extracted content to avoid the need to extract it again if it does not change. By utilising this caching the tool can become even more cost efficient and the potentially expensive cognitive operation can be skipped if the relevant content has not changed.

6.1.2 Automated Model Training

Model training is an important process that should be done regularly to ensure model decay does not affect the model's ability to classify documents. As this is a such a routine process it can be automated to begin after set period, meaning that data scientists can have certainty that the model is always relevant. The benefits of this process are,

- Continual improvement of the model
- Sustained relevance due to alignment to most recent data

- The ability to version and retire models as they decay
- Minimal management overhead as the process is automatic

The choice to use Databricks when building models for use in this tool allows for automated model training to be realised with a very small amount of effort because Databricks notebooks can be invoked by a Data Factory pipeline, that can be set to run on any schedule. Providing the notebook was passed the directory path to the training data it could be run as often as needed and the newly trained model deposited into the data lake for integration to the scoring function. To illustrate this simplicity, an example Data Factory pipeline is included below:

Figure 12: A Data Factory pipeline used to auto-train a classification model

6.1.3 Automated Model Training based on User Input

In addition to automated training, the model could be retrained to improve on specific cases where a bad prediction was made. By isolating documents where a record manager has manually corrected a prediction, further data profiling could be conducted to better understand the features that represent these particular examples. Additionally, if these cases arrive in small numbers then techniques such as SMOTE can be used to synthetically generate examples of the documents so that the model can be trained to classify these records more accurately.

6.1.4 Power BI integration of Knowledge Store

The Azure Cognitive Search Knowledge Store persists documents that have been enriched using AI outside of the search Index. Before the arrival of knowledge store, enriched documents were transitory and could not be visualised after the fact however now the information can be written to both Azure blob and table storage for later analysis. During the research the blob option was used heavily as a way of analysing the features generated from a skillset however a feature that was not utilised is the knowledge stores ability to interact with Power BI. By writing the enriched data into Azure Table storage, analysts can examine the data using the rich capabilities of Power BI and even circulate these reports through the Power BI web service for wider consumption. The below image is an example report built using Azure Cognitive Search Knowledge Store for table storage.

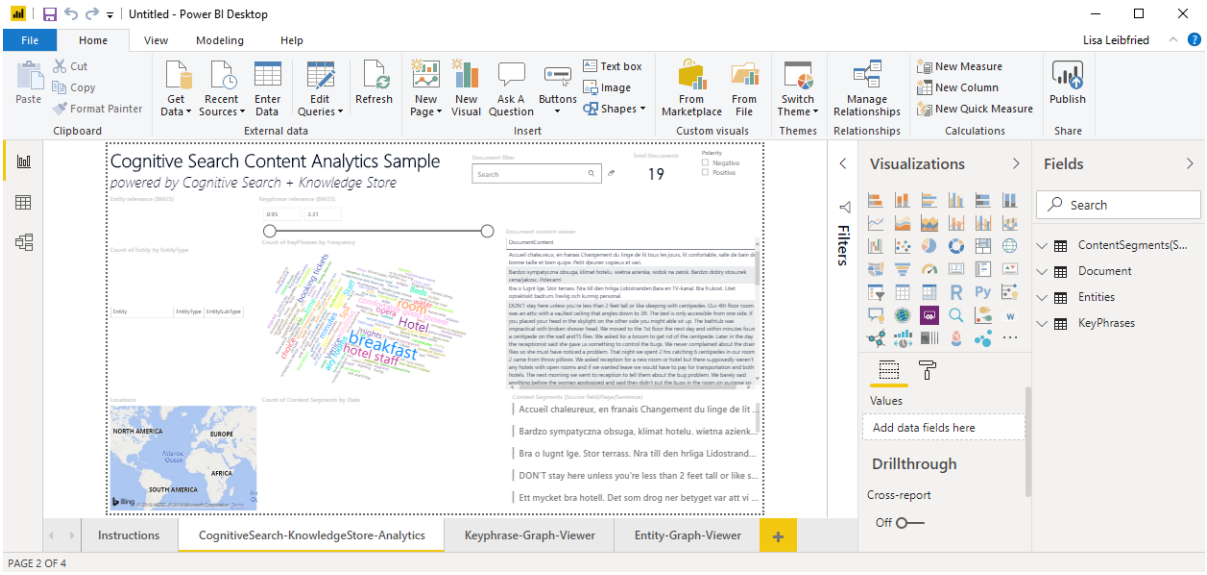


Figure 13: An example of the Power BI Knowledge Store report

6.1.5 Alternative methodologies

Given the limited time for producing a working POC solution, the decision was taken to utilise Microsoft Cognitive Services for entity extraction and key phrase extraction. Not only do these provide quick and accurate results but they also integrate very easily with Azure Cognitive Search. However, these services have a cost associated with them which is larger than the costs of other elements of the tool, whilst also providing limited amounts of configuration. Alternative methods to perform entity and key phrase extraction could be examined to determine if they provide similar if not matching results for a reduced cost or increased configuration options. Also, alternative methods for topic modelling exist and could be examined further. Examples include Bidirectional Encoder Representation from Transformers models (BERT models) which are developed by Google and used to understand search history with a high degree of accuracy.

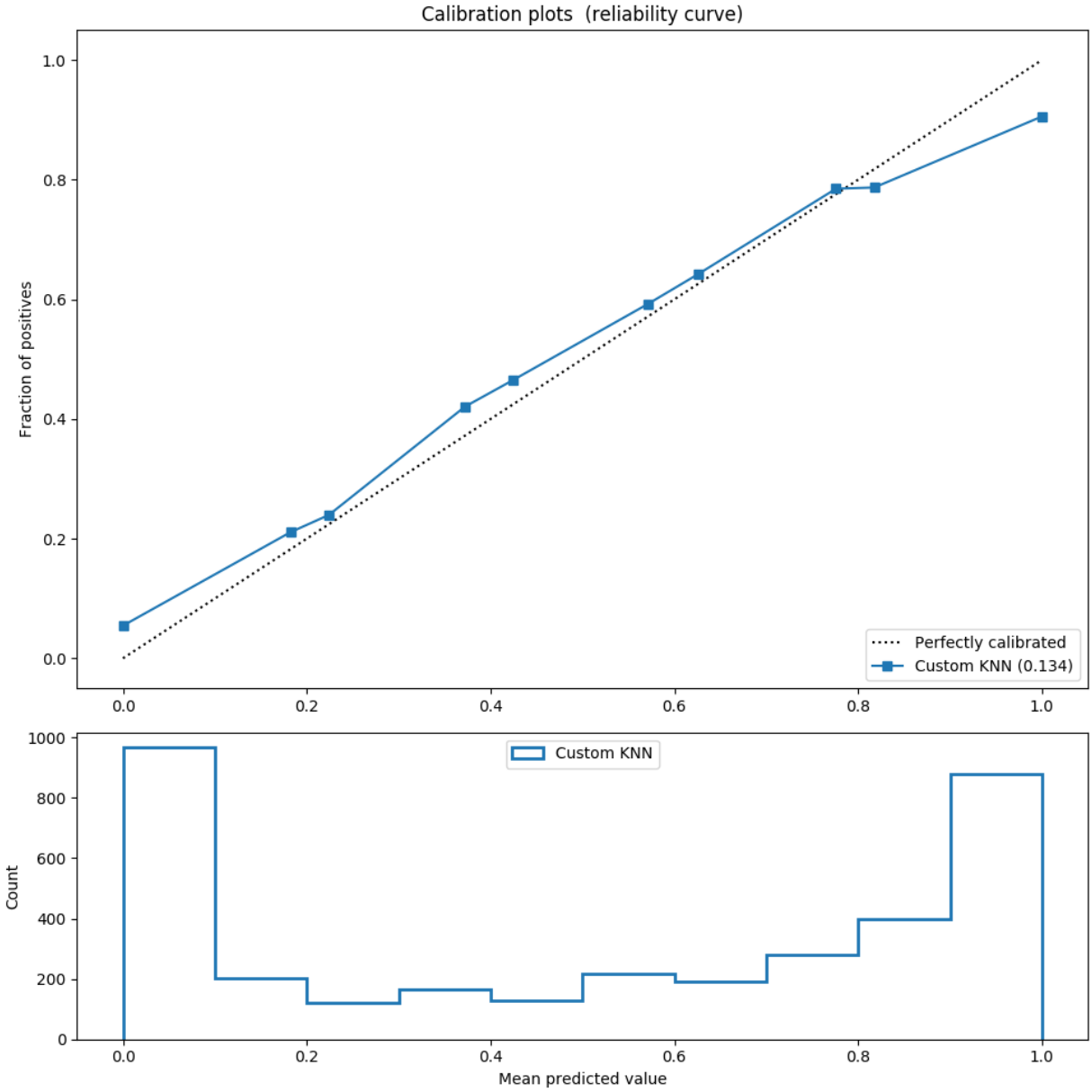
For further information regarding newly released Azure Cognitive Search capabilities, use this link: https://docs.microsoft.com/en-us/azure/search/whats-new

7 Appendices

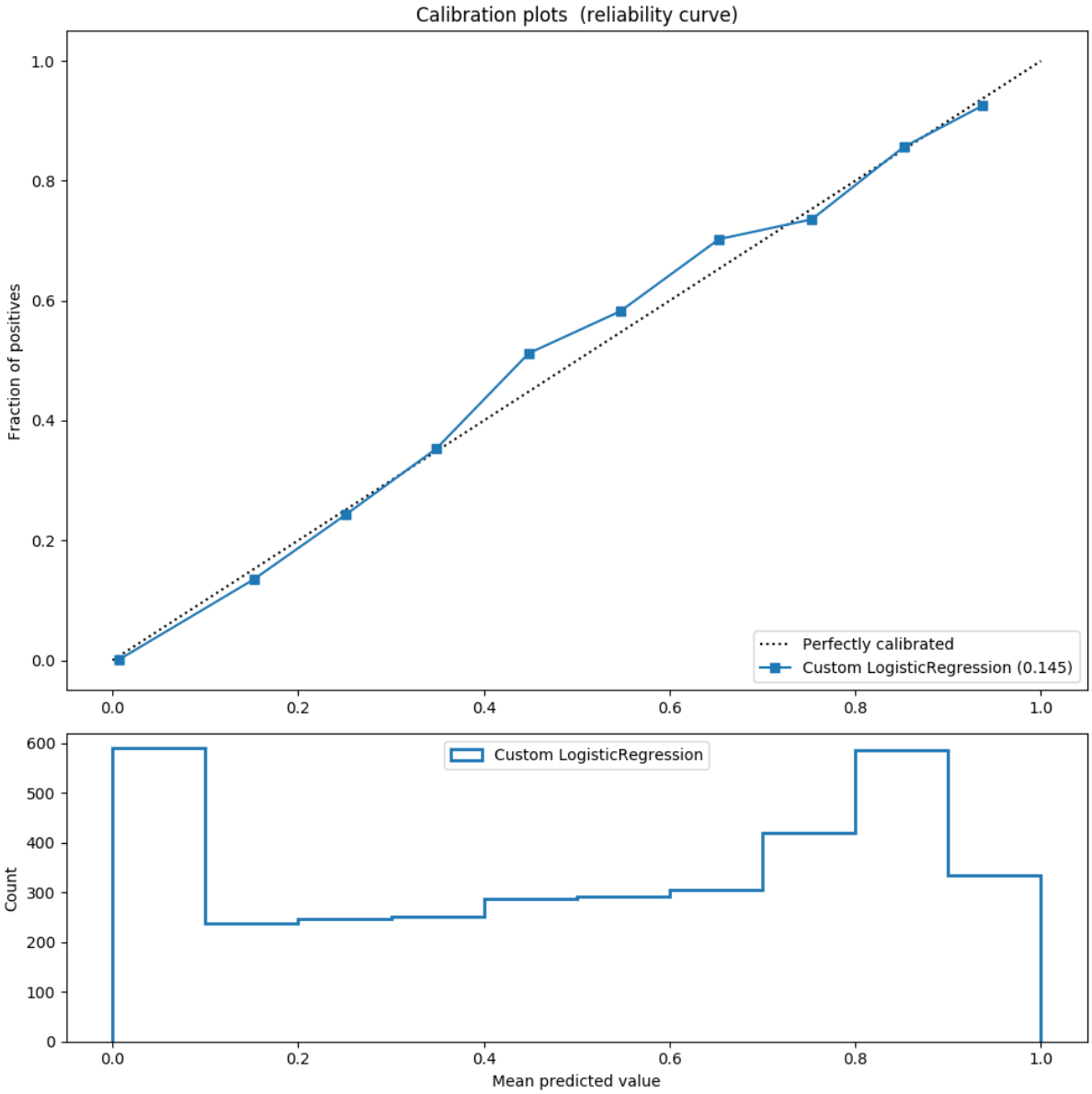
7.1 Calibration Plots

The below images show the calibration plot and mean predicted values for each classification model used in the tool. These plots help data scientists determine the optimum parameters to use for each algorithm.

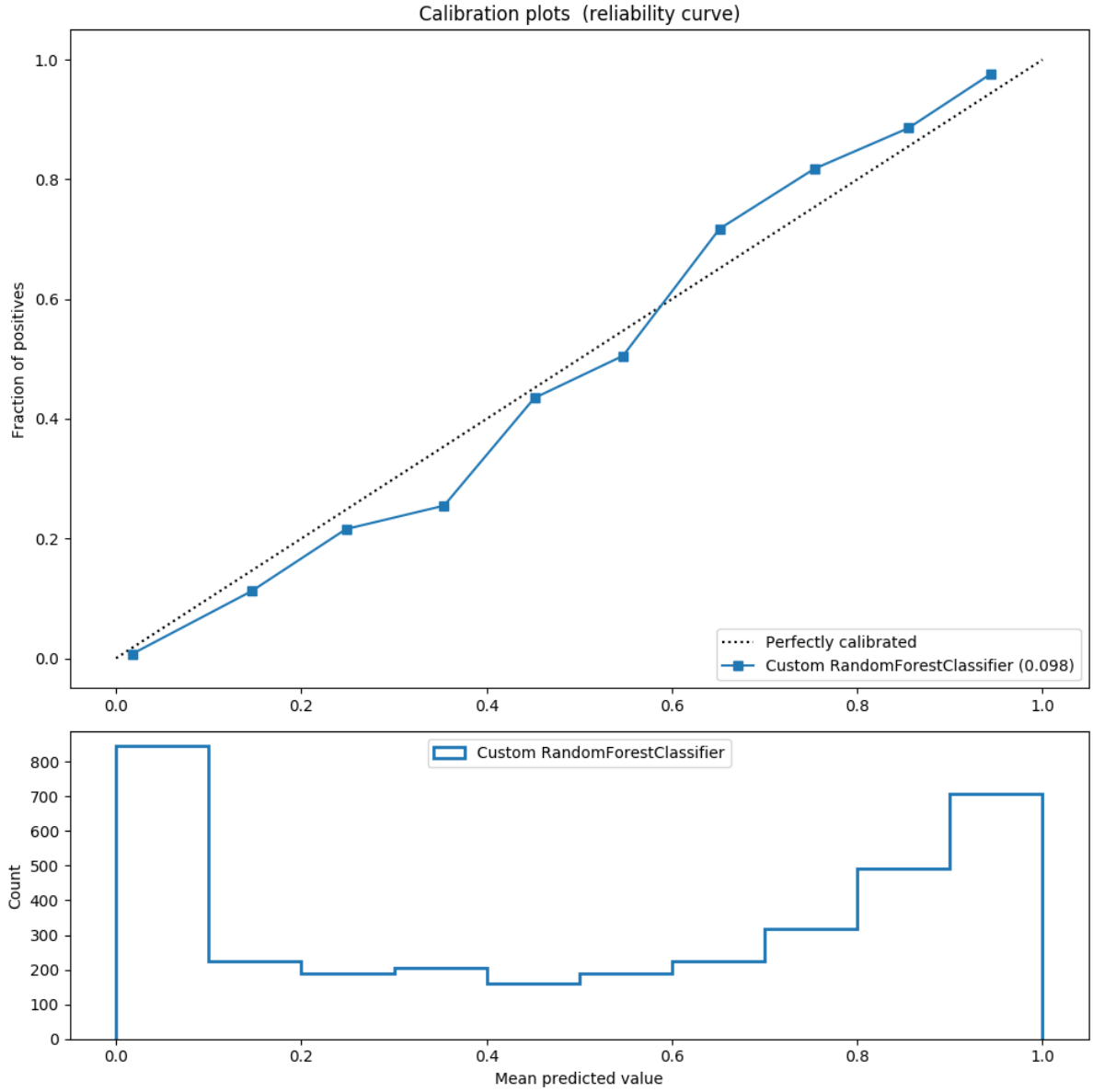
7.1.1 K Nearest Neighbours Calibration Plot



7.1.2 Logistic Regression Calibration Plot



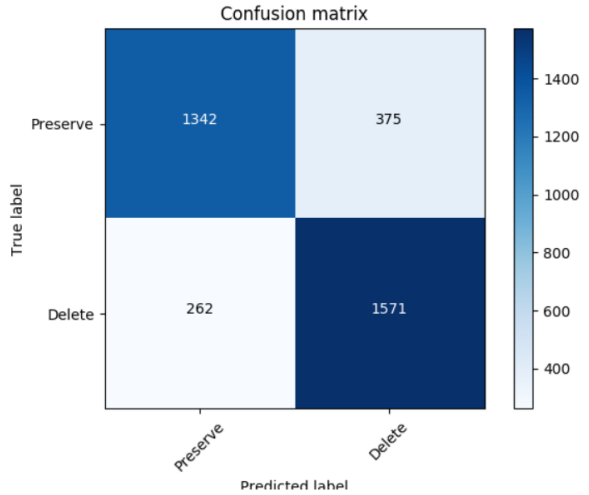
7.1.3 Random Forest Calibration Plot



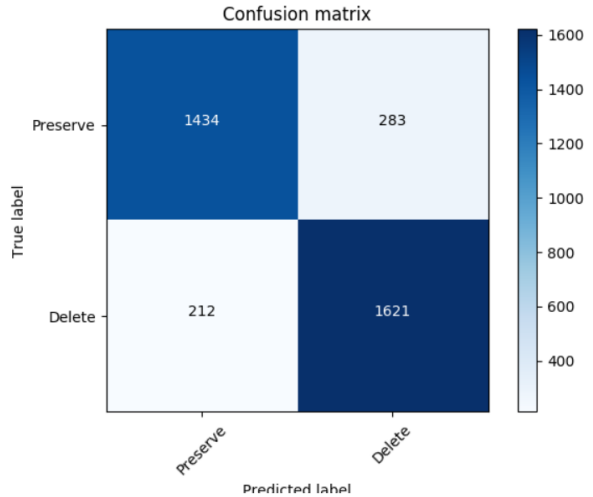
7.2 Confusion Matrices

The below images show the confusion matrices for each classification model used in the tool. These plots help data scientists determine the accuracy for each algorithm.

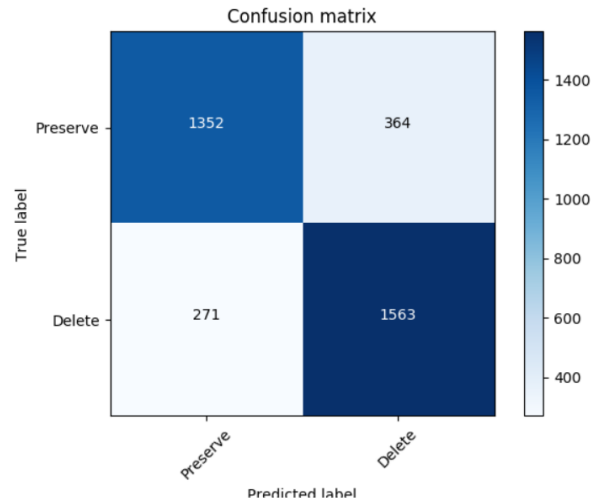
7.2.1 K Nearest Neighbours Confusion Matrix



7.2.2 Logistic Regression Confusion Matrix



7.2.3 Random Forest Confusion Matrix



7.3 K Fold Cross Validation Results

The following result sets show the evaluation metrics for each k fold and the average across the set.

7.3.1 Random Forest K Fold Cross Validation Result

estimator	test_accuracy	test_precision	test_recall	test_f1
RandomForest	0.853873239	0.837801609	0.878425861	0.857632933
RandomForest	0.866549296	0.848930481	0.892480675	0.870161014
RandomForest	0.859105319	0.844444444	0.881236824	0.862448418
RandomForest	0.874251497	0.84882199	0.911454673	0.87902406
RandomForest	0.863332159	0.842035691	0.895291637	0.867847411
Avg	**0.863422302**	**0.844406843**	**0.891777934**	**0.867422767**

7.3.2 Logistic Regression K Fold Cross Validation Results

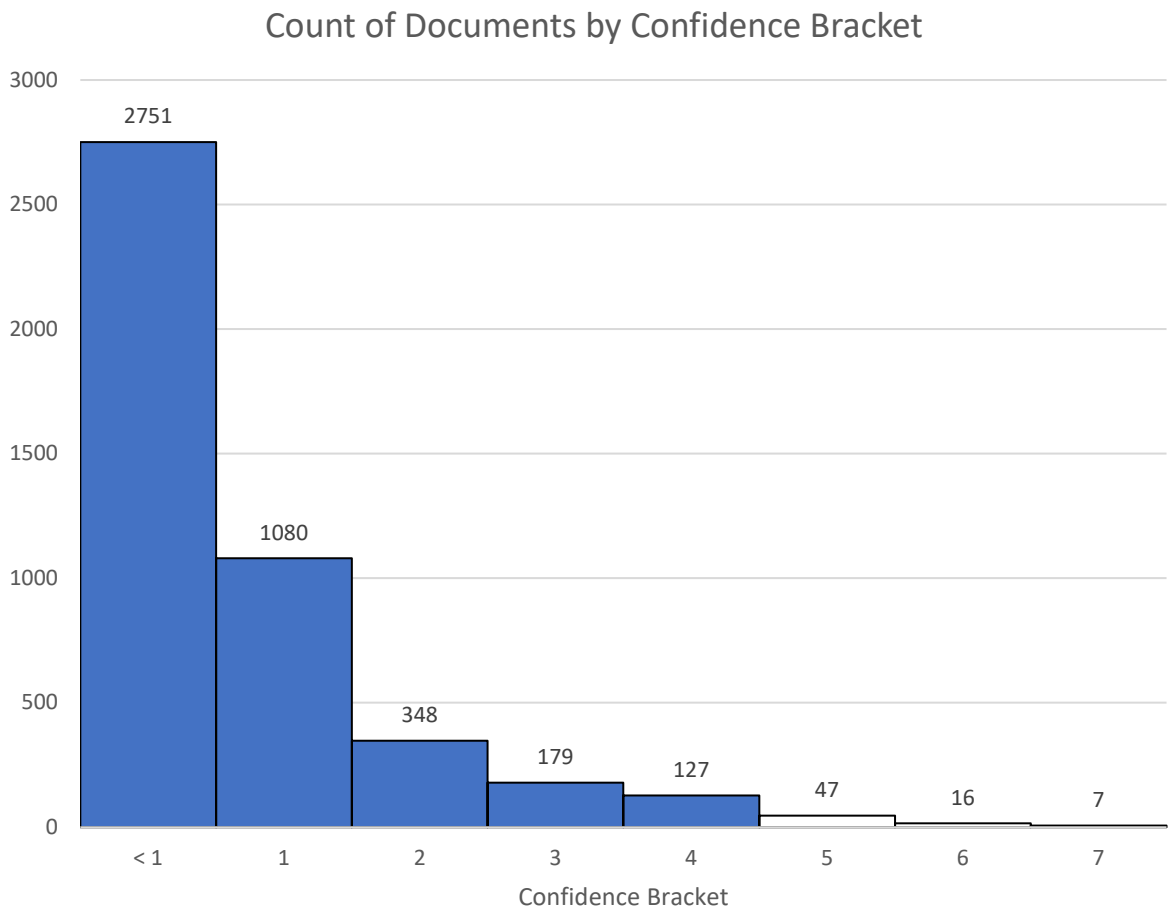
estimator	test_accuracy	test_precision	test_recall	test_f1
LogisticRegression	0.765492958	0.748522653	0.801124385	0.773930754
LogisticRegression	0.794366197	0.77653263	0.827828531	0.801360544
LogisticRegression	0.791475872	0.771746239	0.829234013	0.799457995
LogisticRegression	0.80873547	0.790237467	0.841883345	0.81524328
LogisticRegression	0.796054949	0.780585106	0.825017569	0.802186539
Avg	**0.791225089**	**0.773524819**	**0.825017569**	**0.798435822**

7.3.3 K Nearest Neighbours K Fold Cross Validation Results

estimator	test_accuracy	test_precision	test_recall	test_f1
Kneighbours	0.808098592	0.79305741	0.834855938	0.813420062
Kneighbours	0.81971831	0.802256138	0.849613493	0.825255973
Kneighbours	0.819302571	0.797385621	0.85734364	0.826278361
Kneighbours	0.835153223	0.813938199	0.869992973	0.841032609
Kneighbours	0.815427968	0.791693705	0.85734364	0.823211876
Avg	**0.819540133**	**0.799666215**	**0.853829937**	**0.825839776**

7.4 Unlabelled prediction histogram

A histogram showing the distribution of confidences for document selection based on unlabelled data


Count of Documents by Confidence Bracket
3000
2500
2000
1500
1000
500
0
2751
1080
348
179
127
47
16
7
< 1
1
2
3
4
5
6
7
Confidence Bracket