CHAPTER 1

Design

In this chapter, we explain the methodology used to fill the research gap identified in Chapter **??**, thereby producing a system that identifies high-potential startup companies that are likely to receive additional funding or a exit in a given forecast window. Figure 1.1 depicts the overall system architecture, which will be described in the following chapter. We will evaluate the performance of this system in the next chapter, Chapter **??**.

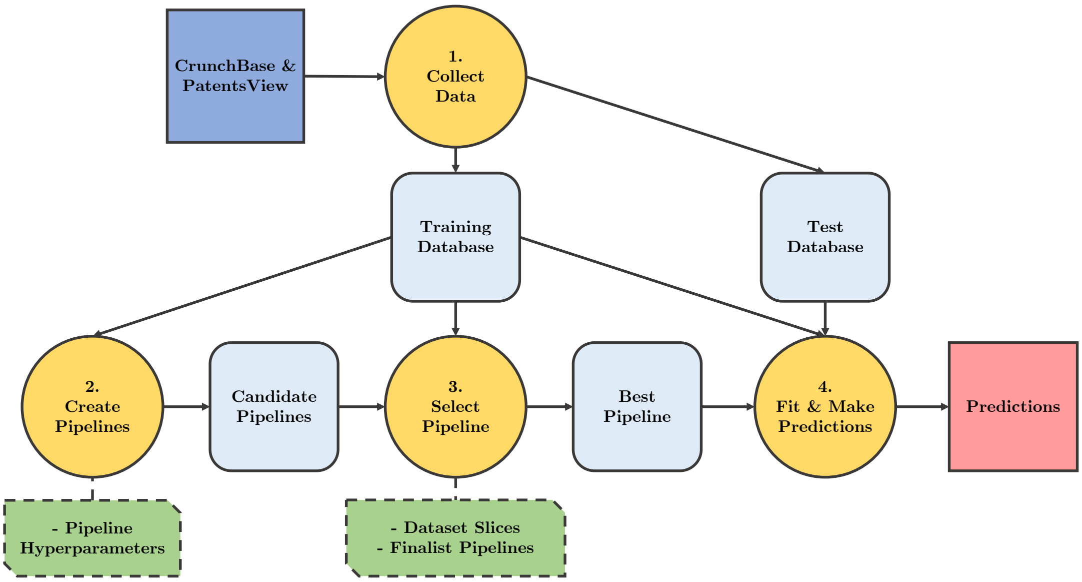


Figure 1.1: An overview of the system architecture proposed by this project, structured in four stages: data collection, pipeline creation, pipeline selection and prediction.

1. Dataset Preparation. Our primary data source was the CrunchBase online database, which we supplemented with patent filing data from PatentsView. We collected two database dumps from CrunchBase in September 2016 and

April 2017, for training and testing respectively. The database dumps were in the format of CSV files which we imported into a relational database (SQLite) and performed aggregation queries to create features suitable for classification. We then performed screening based on each company's developmental stage and age to ensure only relevant companies were included in the dataset. Finally, we explored the dataset and identified issues of sparsity, long-tailed distributions, imbalanced feature ranges, and non-orthogonality.

2. Pipeline Creation. We developed a processing pipeline framework that seeks to address the dataset issues identified during data collection. Our pipeline is based on the popular Python machine learning library Scikit-learn [**pedregosa2011**]. Pre-processing steps include imputation, transformation, scaling and feature extraction. Each pre-processing step has hyperparameters that can be tuned (e.g. imputation strategy, number of components to extract) that affect the pipeline's classification performance. We also tested a number of common classification algorithms and their hyperparameters, selected from our literature review in Chapter **??**. We performed a search across the pipeline's hyperparameters to generate candidate pipelines. The hyperparameters that we found to have the most significant effect on the final performance of the pipelines were related to the classification algorithms.

3. Pipeline Selection. Our system evaluates and ranks the candidate pipelines and tests the best pipelines (finalist pipelines) over pseudo-historical datasets. This process ensures we select pipelines that are robust with respect to time. We prepared the dataset slices using a technique that filters records by their timestamps, effectively recreating historical datasets. We use Area under Precision-Recall (PR) Curve as our evaluation metric. We aggregate the results for each finalist pipeline across these dataset slices and rank the finalist pipelines on their overall performance, so we can select the best pipeline for further evaluation. We don't observe significant variance in the pipelines on aggregate against the dataset slices, but there is variance within the individual pipelines. Our results suggest that it is optimal to evaluate the top 3-5 candidate pipelines in this manner.

4. Model Fit and Prediction. Finally, our system applies the best pipeline to a training dataset to produce a fitted model. The model is then applied to a feature vector from a held-out test database, which generates a set of predictions which could, in practice, then be used by Venture Capital (VC) firms. We evaluate the accuracy of the models produced by our system with respect to a number of variables in the next chapter, Chapter **??**.

1.1 Dataset Preparation

In the previous chapter, we reviewed the literature concerning data sources for entrepreneurship and Venture Capital (VC) investment. We concluded the most promising primary data sources for this project are CrunchBase and AngelList, for their size, comprehensiveness and ease of access. We suggested PatentsView (the online database of the US Patent Office) could be a useful secondary data source for structural capital features. In the following sections, we first discuss how we collected data from CrunchBase and PatentsView, converted the relational databases into a format suitable for machine learning, performed preliminary screening to ensure we only included relevant companies. This process is depicted in Figure 1.2. Following our description of this process we describe the results of exploratory analysis on our dataset and identify dataset issues which will be addressed in later steps.

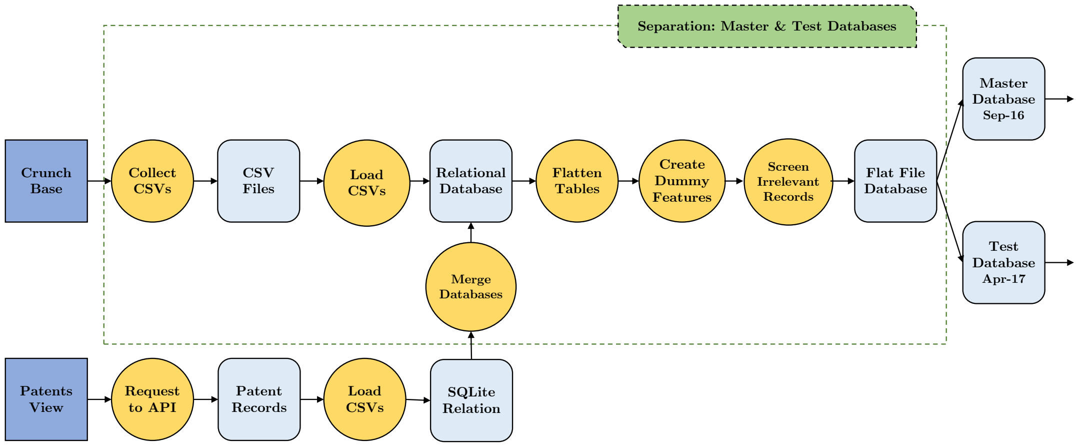


Figure 1.2: Data collection overview.

1.1.1 Data Collection

1.1.1.1 CrunchBase

CrunchBase is an online, crowd-sourced repository of startup companies, individuals and investors with a focus on US high-tech sectors. CrunchBase is freely accessible for browsing but requires licenses to filter the dataset, use the API, and download Microsoft Excel and CSV-formatted dumps. For the purposes of this project, we were granted an Academic License. CrunchBase provides database access in a few formats that offer trade-offs in terms of accessibility and comprehensiveness. We intended to use CrunchBase's API because it provides the most comprehensive access to their database. We developed a collector that downloaded a daily list of updated API endpoints from CrunchBase and queried nodes it needed to update. CrunchBase's API provides JSON-formatted responses which the program recursively parsed and stored into a relational database. However, due to the time constraints of this research project, we abandoned this data collection method. CrunchBase also provides CSV-formatted dumps of their key endpoints (e.g. organizations, people, funding rounds). We downloaded two CSV-formatted dumps from Crunchbase on 09 September 2016 and 04 April 2017 which we loaded into relational databases (see Appendix **??** for the full database schema).

1.1.1.2 PatentsView

In 2015, the United States Patent and Trademark Office (USPTO) launched PatentsView, a free public API to allow programmatic access to their database. PatentsView holds over 12 million patent filings from 1976 onwards [1]. The database provides comprehensive information on patents, their inventors, their organisations, and locations. We collected the patent filing records of each company in the primary database, focusing on information relating to dates, citations, and patent types. We matched the data sources on standardised company names (removing common suffixes, punctuation etc.) and using normalised Levenshtein distances. Although approximate matching introduces error, the volume of companies in the database is too high to be matched manually and there are no other identifying records. We stored the PatentsView data in a relation which we merged into our master and test databases.

1.1.2 Dataset Manipulation

To prepare the dataset for machine learning, we first flattened the relational database into a single file using SQL aggregation queries. We aggregated each relevant relation in turn, grouping by Company ID and then combined each aggregated table using left outer joins. Following this process, we used Python to convert tuples (e.g. Round Type and Round Date) and lists (e.g. Round Types) into dummy variables.

We performed preliminary screening on the primary dataset (N = 425,934) to ensure it only included relevant companies. We were interested in removing traditional, non-startup businesses from the dataset (e.g. consulting firms, companies that will not take VC funding etc.). To do this, we explored two factors for each company: developmental stage and age. By developmental stage, we primarily refer to external funding milestones. These stages are associated with shifts in a startup company's functions and objectives and we also expect them to correlate with company age. Our dataset as grouped by startup developmental stage is depicted in Figure 1.3.

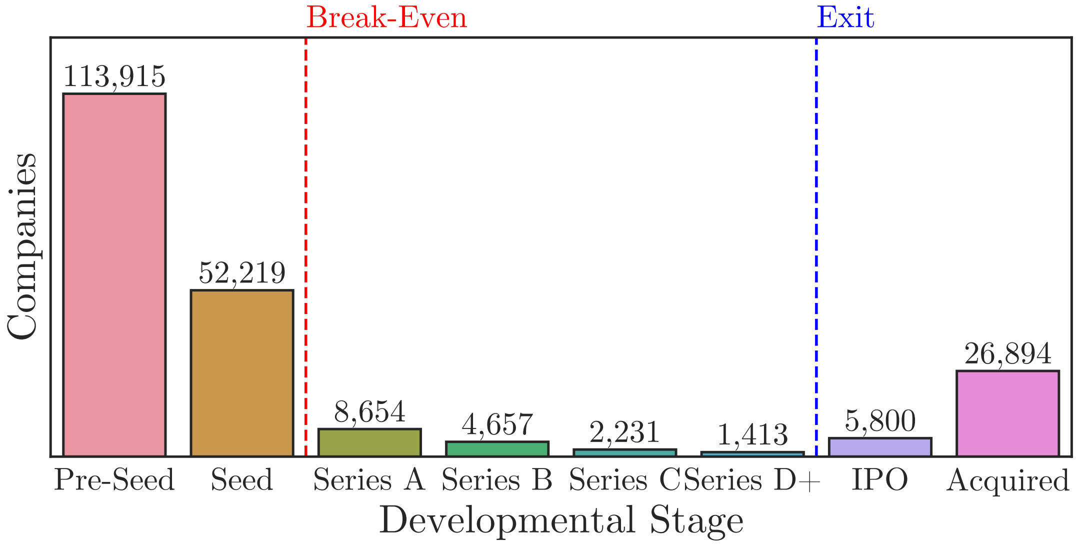


Figure 1.3: Idealised startup development lifecycle (adapted, []) with company counts from the master dataset (c. September 2016). Red line represents profitability over time. The chart is divided into three periods: an early period of unprofitability ("Valley of Death") where seed capital supports the business, a period of growth sustained by rounds of venture capital, and a transition to stability and mature capital markets.

After attempting to place the companies into development stages we are left with a large group of companies (the majority of the dataset) that have not raised funding and so can not be classified on that basis. We assume that companies that have not raised funding fall into two groups - those that intend to raise funding but have not had time to yet, and those that have chosen not to pursue funding and are unlikely to do so. We separated these groups by applying a cutoff equal to the 90th percentile of the age of companies in the Seed category, and excluded the older group from further analyses (N = 227,162, 53.3%). As we are only interested in companies that could theoretically seek investment, we also excluded Closed, Acquired and IPO groups from further analyses (N = 35,973, 8.4%).

Figure1.4 depicts the ages of companies in the master dataset, grouped by developmental stage. As we expected, there is a strong relationship between company stage and company age. Most pre-Series A companies are under five years old, and the majority of Series D+ funded companies are under 10 years old and the 75th percentile is at 15 years old. On this basis, we excluded companies that are over the 75th percentile of the age of companies in the Series D+ category (N =9,756, 2.2%). Overall, our preliminary screening steps reduced the dataset from 425,934 companies to 153,043 companies, a reduction of 64.1%.

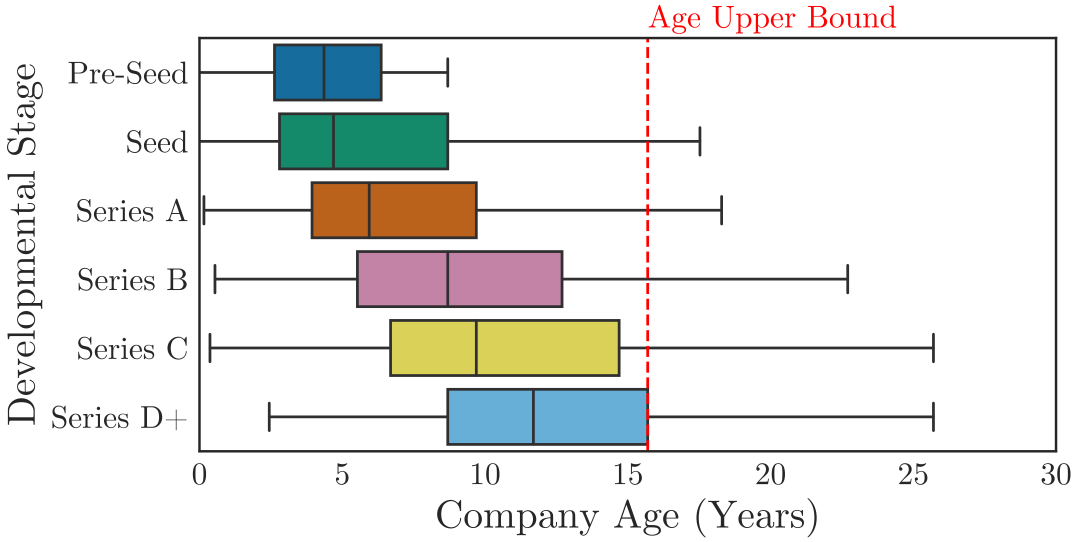


Figure 1.4: Company ages in years grouped by developmental stage. While there is significant variability in age for each developmental stage, there is a broad positive relationship between age and developmental stage. The dashed blue line represents the 75th percentile of the age of companies in the Series D+ category (15.7 years).

1.1.3 Exploratory Analysis

1.1.3.1 Descriptive Statistics

Table 1.1 presents the descriptive statistics for the dataset. The dataset is heavily skewed towards New companies (i.e. companies that were recently founded and have not raised any type of funding yet, 68.9%). These companies have very few available features in comparison to companies at later developmental stages. We will investigate the impact of this sparsity on our predictions in Chapter **??**. We are presented with a fairly heterogenous dataset, the interquartile ranges imply significant variability in all measures. We do not believe that this implies that the data has not been cleaned effectively, but rather, reflects that startup companies naturally vary significantly in their traits.

	Count	Age (Years)		Funding Raised (USD, millions)		Funding Rounds (N)		Available Features (N)	
Stage	N	Median	IQR	Median	IQR	Median	IQR	Median	IQR
Pre-Seed	113,915	4.36	3.74	0.00	0.00	0.0	0.0	33	127
Seed	38,942	4.66	4.00	0.25	1.30	1.0	1.0	185	127
Series A	6,615	5.69	5.01	4.40	7.91	2.0	1.0	246	117
Series B	3,342	7.61	5.92	14.89	22.03	2.0	2.0	261	87
Series C	1,610	8.70	5.52	35.29	45.88	3.0	2.0	311	80
Series D+	998	9.70	5.01	74.39	90.30	5.0	3.0	324	69
Total	165,422	4.69	4.00	0.0	3.97	1.0	1.0	98	145

Table 1.1: Descriptive statistics grouped by developmental stage. Source: Master dataset (c. Sep-16).

CrunchBase's approach to industry classification is simplistic compared to other classification schemes (e.g., USSIC, NAICS, VentureSource) which generally have an industry hierarchy with tiers for broad industry sectors and subsectors providing further granularity. As a result, CrunchBase class labels include over represented and vague classes (e.g., "Software", "Internet Services") which could describe the majority of companies included in the database. In fact, "Software" and "Internet Services" account for 16.4% and 13.4% of all companies in the dataset respectively (see Figure 1.5). Despite these vague class labels, it is clear the dataset skews towards high technology startups, as opposed to biomedical, agricultural, or other technologies.

1.1.3.2 Sparsity

First, we explored missing data in the dataset. We expected the dataset to be highly sparse because it primarily came from CrunchBase, a crowd-sourced database. As profiles are entered into CrunchBase piece-meal, it is not clear at face-value whether data (e.g. records of funding rounds) is missing or didn't occur. Figure 1.6 displays the distribution of missing data in the dataset, with respect to each feature and each feature. The multi-modal peaks of both distributions suggest that missing data across certain groups of features may be correlated with each other (e.g. all features derived from funding rounds).

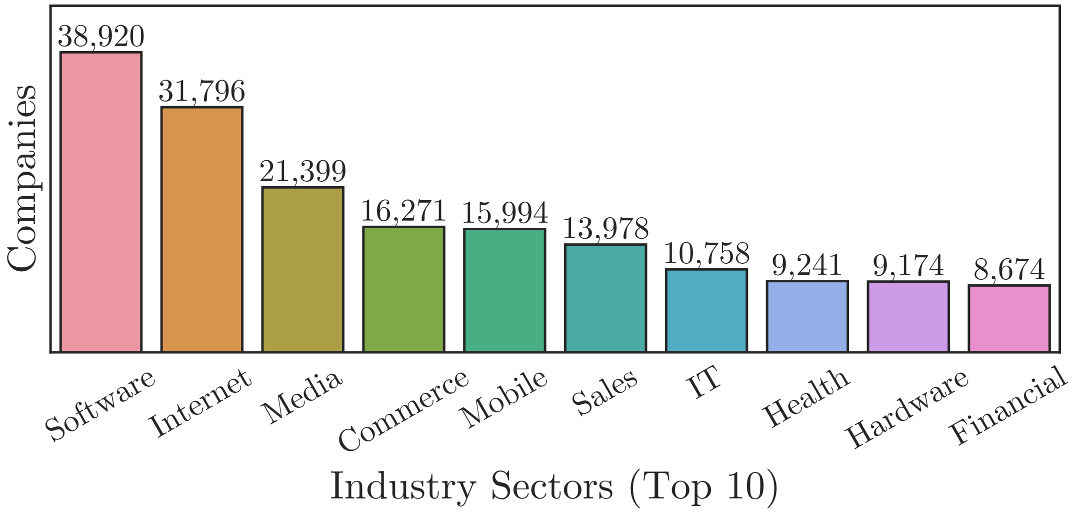


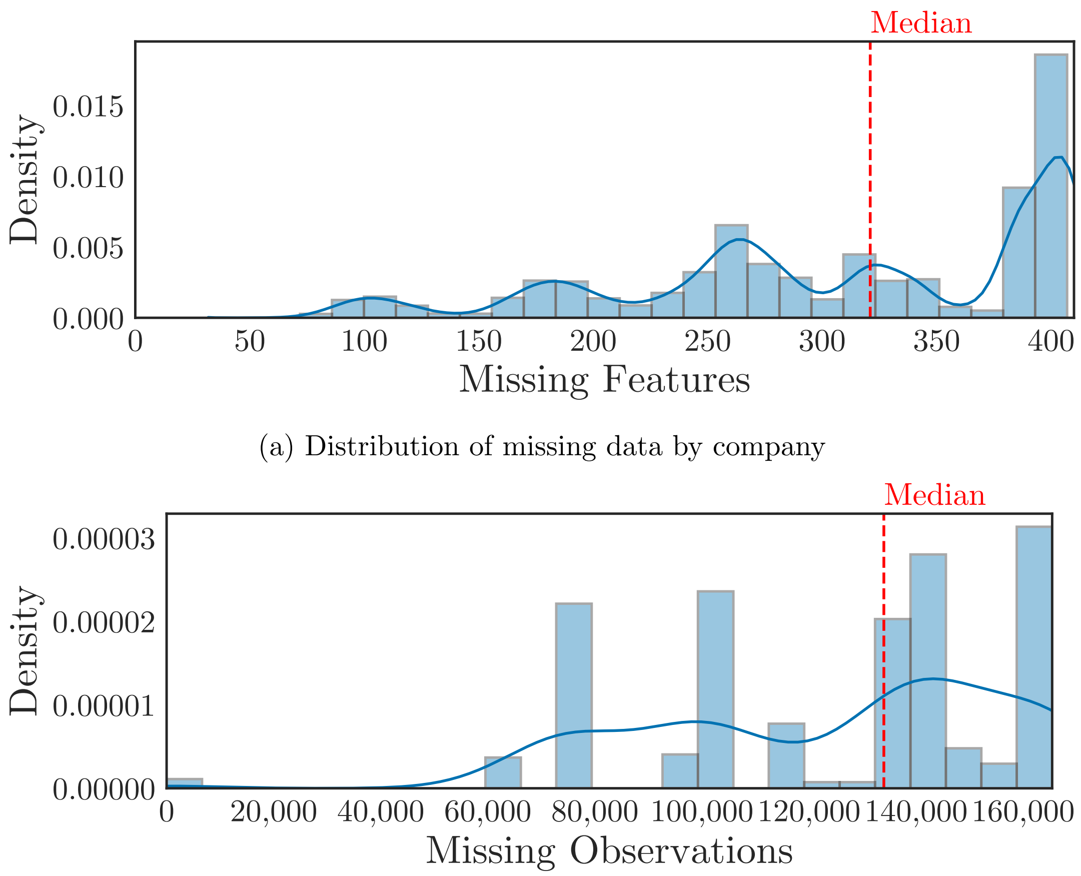
Figure 1.5: Companies grouped by industry sector. The 10 most common sectors are displayed. Source: Master dataset (c. September 2016).

1.1.3.3 Normality

Next, we explored the distributions of features. Figure 1.7 shows the skewness and kurtosis of the features in our dataset. A feature is considered horizontally symmetrical if it has a skewness of 0 and generally considered highly skewed if its absolute skewness is above 1 [**bulmer1979**]. The vast majority of our features are more skewed than this cutoff. Kurtosis is a measure of the distribution of variance in a feature. We use Fisher's measure of kurtosis, which has a normal value of 0. Our dataset has consistently higher kurtosis than normal which suggests that we have many extreme values in our dataset. These results in concert suggest that our dataset has many features that are positively skewed with long-tail distributions. This is intuitively what we might expect for features like "amount of funding raised".

1.1.3.4 Scale

Next, we explored the scaling and range of each of our features. Figure 1.8 shows the Interquartile Range (IQR) of each feature (transformed by log1p for ease of viewing). The distribution is extremely skewed, which shows that our features have very different scaling. This may be an issue for our machine learning estimators and feature extractors, so we address this by applying a scaler in our pipeline.



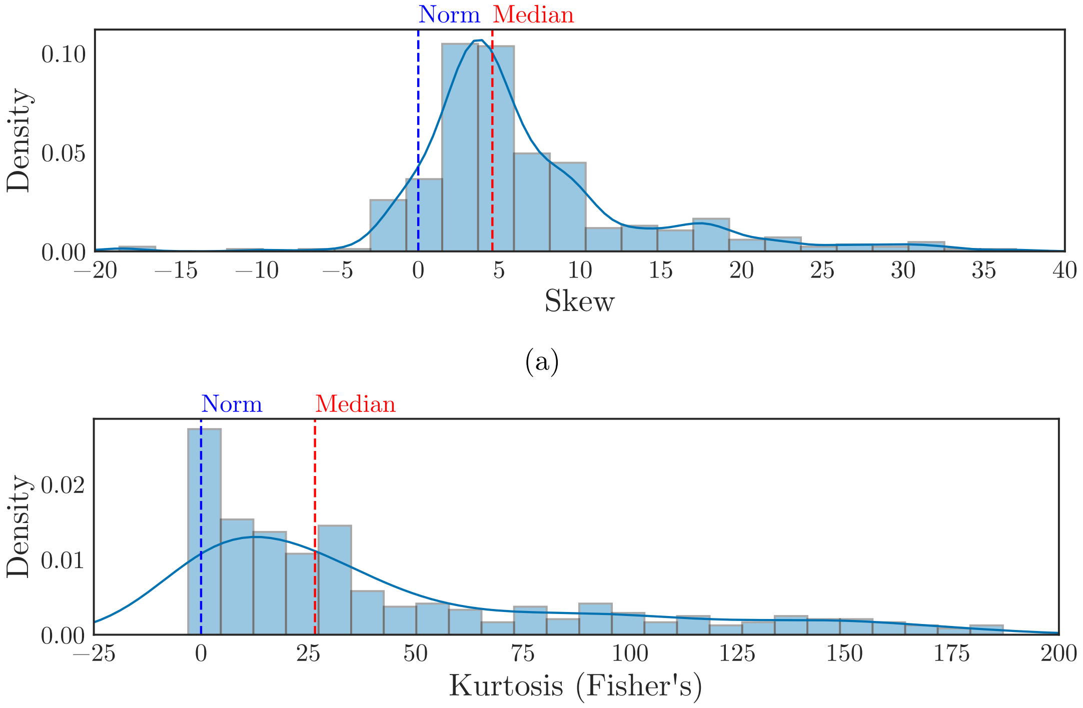
(a) Distribution of missing data by company

(b) Distribution of missing data by feature

Figure 1.6: Distribution of missing data in master dataset (c. September 2016).

1.1.3.5 Orthogonality

Finally, we explored the orthogonality of our features: the extent to which the variance of our features is unrelated. This is a less straight-forward measure. We explore pair-wise inter-correlations between our features and evaluate how many of the inter-correlations are above a particular correlation cutoff, as depicted in Figure 1.9. We use two correlation metrics: Pearson and Spearman. Pearson is more commonly used but Spearman is a ranked metric and may more accurately reflect our non-normal feature distributions. Although most features have relatively low inter-correlations (6̃0% below 0.2) there are still a considerable number that are highly correlated, so it might be efficient to remove these features using an unsupervised feature extracter prior to estimation.



(a)

(b)

Figure 1.7: Distribution of features in master dataset (c. September 2016).

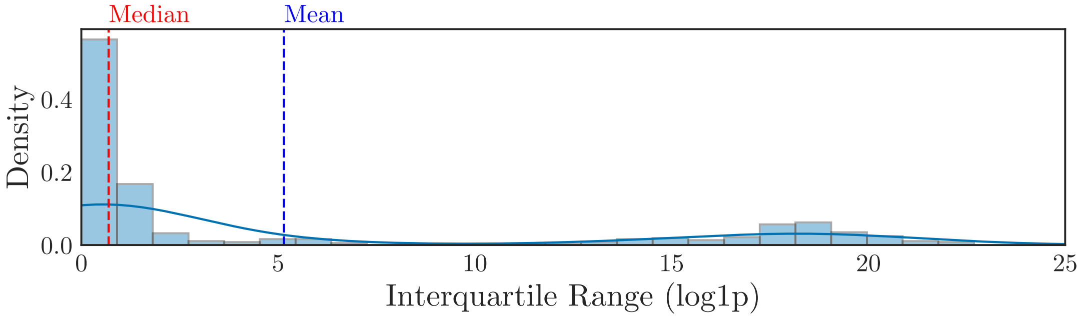


Figure 1.8: Distribution of interquartile ranges (transformed by log1p) in master dataset (c. September 2016).

1.2 Pipeline Creation

We developed a classification pipeline using the popular Python-based machine learning library Scikit-learn [**pedregosa2011**]. The classification pipeline construct allows us to easily search across hyperparameters at each step in the

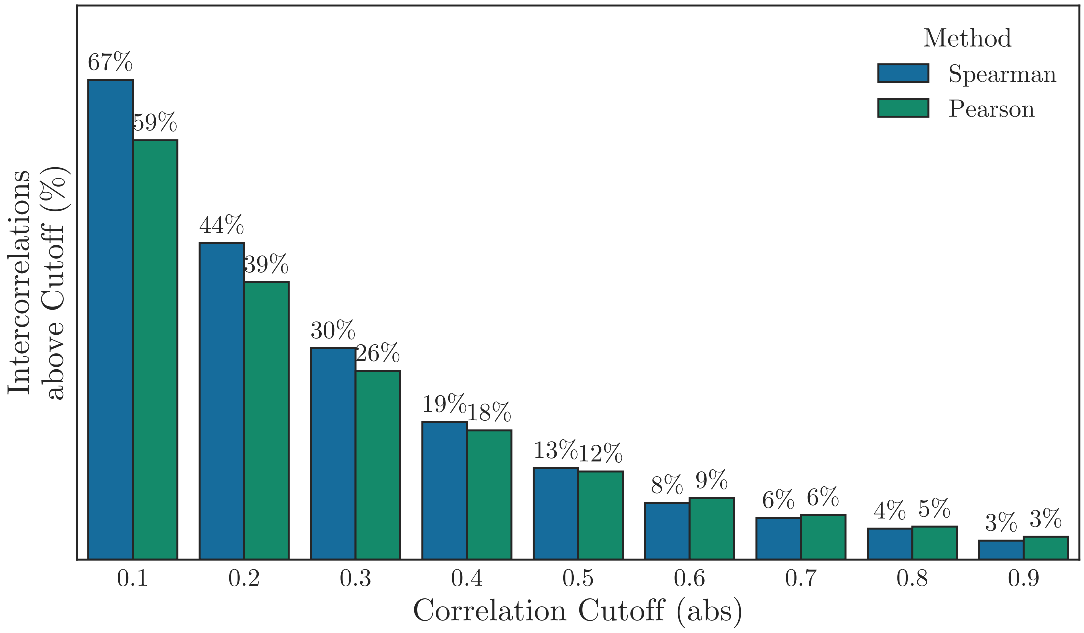


Figure 1.9: Distribution of intercorrelations in master dataset (c. September 2016).

pipeline (see Appendix **??** for hyperparameter list). The following sections explore the testing of each hyperparameter decision, and the selection of primary classifiers for the following steps. This process is depicted in Figure 1.10.

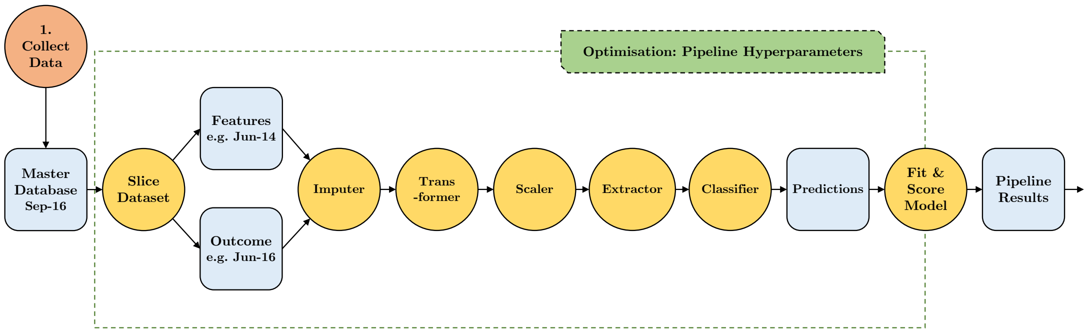


Figure 1.10: Pipeline creation overview.

1.2.1 Imputation

After reviewing the distribution of missing data, we decided to perform further investigation into imputation methods. Common imputation strategies include replacing missing values with the mean, median or mode of each feature. Figure 1.11 shows the distribution of mean, median and modes for each feature in the dataset. For the majority of features, all three measures of central tendency are equal to zero. This resolves the issue of distinguishing missing data from negative observations because, following imputation, all of these data points will map to zero. Figure 1.12 shows the receiver-operating characteristics of the different imputation strategies. As expected, all three imputation strategies produce similar results (within the margin of error).

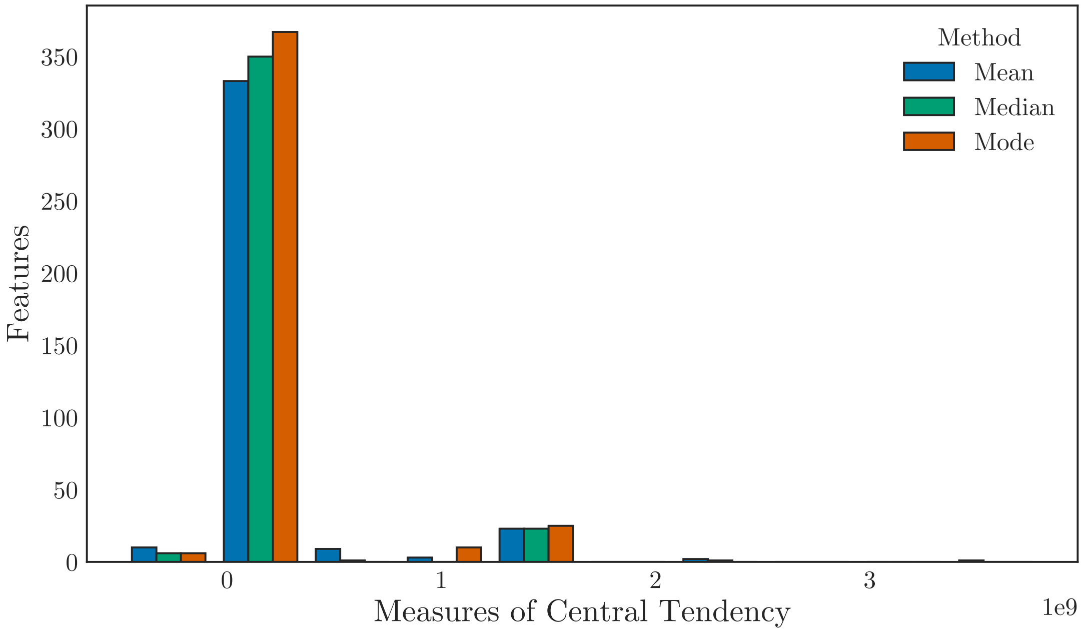


Figure 1.11: Distribution of measures of central tendency (mean, median and mode) in master dataset (c. September 2016).

1.2.2 Transformation

While the classification algorithms we identified in the previous chapter are relatively robust to violations of normality, it may be beneficial to transform the data if the feature distributions are extreme. Table 1.13 shows one of the key features, Total Funding Raised, under a number of different transformations. Like many features in our dataset, the distribution of Total Funding Raised is highly skewed. The log transformation reduces this skew (a normal distribution of non-zero values is apparent) and square root transformation also reduces this skew (to a lesser extent). The impact of these transformations is reduced by the

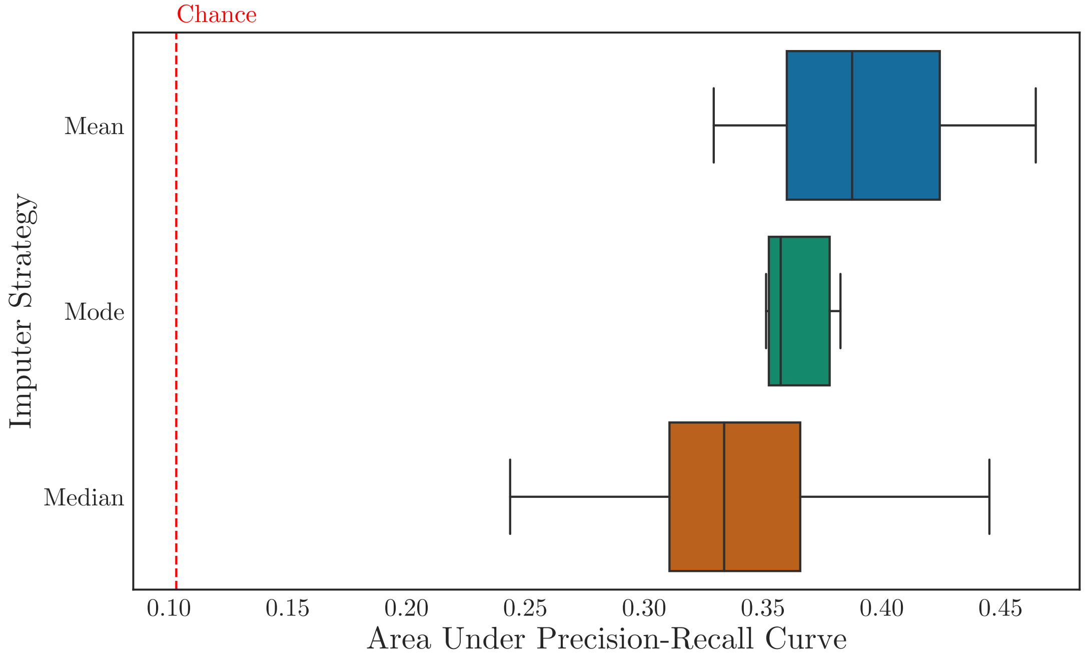


Figure 1.12: Area under Receiver Operating Characteristic (ROC) for different imputation strategies. Imputation strategies include replacing missing values with the most frequent (mode), median and mean value of each respective feature. Results presented are aggregated from hyperparameter optimisation performed over entire classification pipeline (including all classifiers).Source: Features (Apr-12) and labels (Apr-14, 2 year forecast window) derived from Master dataset (c. Sep-16).

extent of their zero-inflation. However, it is still reasonable to expect both of these transformations to improve the classification accuracy. Figure 1.14 shows the ROC of these different transformation functions. Both functions provide a small performance improvement, with the square root function narrowly best.

1.2.3 Scaling

Standardisation of datasets is a common requirement for many feature extraction methods and machine learning estimators. Sci-kit learn provides three primary scaling functions: StandardScaler, RobustScaler and MinMaxScaler. RobustScaler is intended to alleviate the effect of outliers while MinMaxScaler is intended to preserve zero entries in sparse data - both of these are relevant properties for the dataset. Figure 1.15 shows the receiver-operating characteristics of

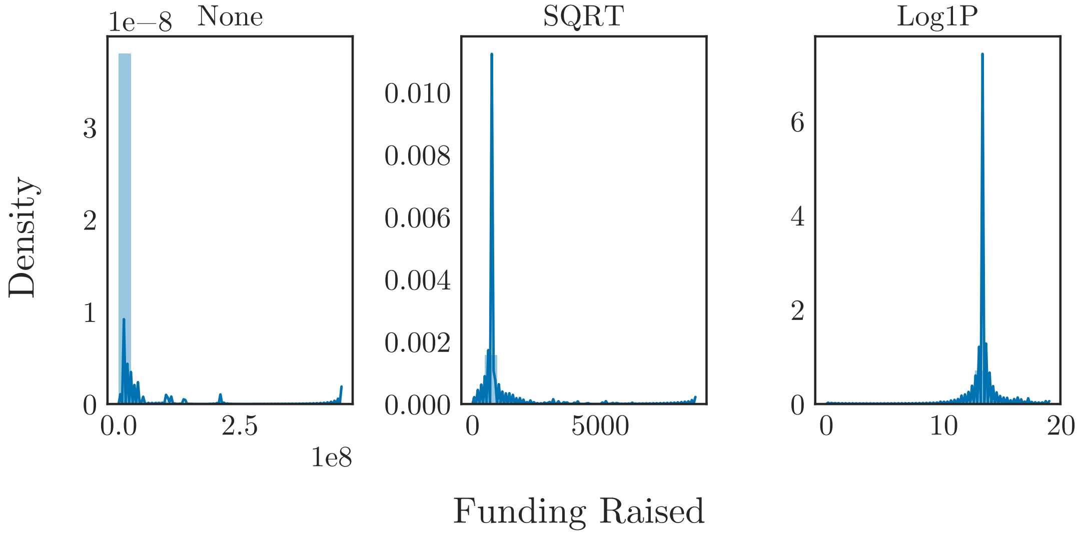


Figure 1.13

the different scaling functions. MinMaxScaler and RobustScaler actually underperform the null condition while StandardScaler only performs on par with the null condition. This is unexpected but may be caused by the previously applied transformations.

1.2.4 Extraction

Feature extraction reduces high-dimensional data into lower-dimensional data in such a way that maximises the variance of the data. The most common approach to dimensionality reduction is Principal Component Analysis (PCA), which constructs orthogonal eigenvectors (components). The magnitude of each eigenvector (its eigenvalue) is displayed in Figure 1.16. The majority of explained variance is captured in the first 10 components, and the Eigenvalues drop below 1 by 100 components - this suggests that these are reasonable values for further hyperparameter search. Figure 1.17 shows the ROC for different numbers of extracted components. All curves produce similar classification results (within margin of error) which implies that we should extract between $1 - 20$ components because it will provide us with more efficient computation.

While PCA is efficient at reducing features, the resultant components are not interpretable. Similarly, individual analysis of 400+ features is difficult to interpret. A compromise is to group the features using the conceptual framework we developed earlier from the literature review. The grouping approach

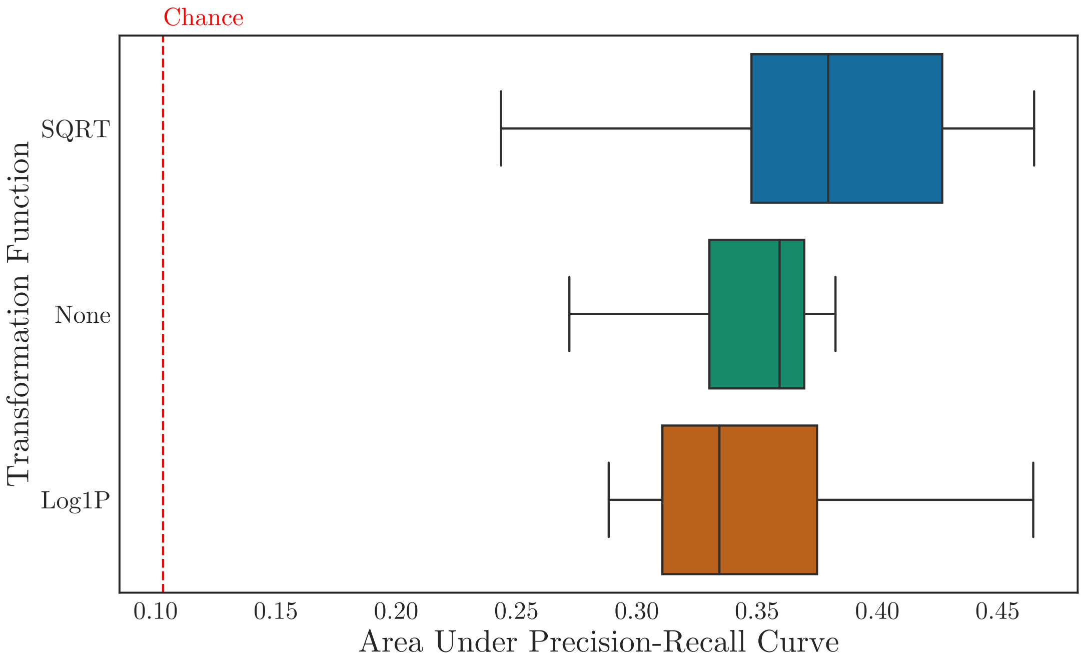


Figure 1.14: Area under ROC for different transformation functions. Transformations include: None (identity transformation), Log1p (natural logarithm of one plus the input array, element-wise), and Sqrt (the square root of the input array, element-wise). Results presented are aggregated from hyperparameter optimisation performed over entire classification pipeline (including all classifiers).Source: Features (Apr-12) and labels (Apr-14, 2 year forecast window) derived from Master dataset (c. Sep-16).

applied weights to each individual feature that optimised the inter-correlations within each group. Given the highly skewed features, we use Spearman correlation which is robust to skewness because it is based on ranking. Figure 1.18 displays the inter-correlations between each factor from the proposed conceptual framework. As we would expect, 'investors' and 'funding' features are highly correlated. While 'investors' attempts to capture the influence of previous investors, it also captures features like the size of an investor's past investments, which would likely correlate with the size of the investment they made in the target company. Interestingly, 'founders' features are positively correlated with all other features except for 'advisors' features which are negatively correlated with all other feature groups.

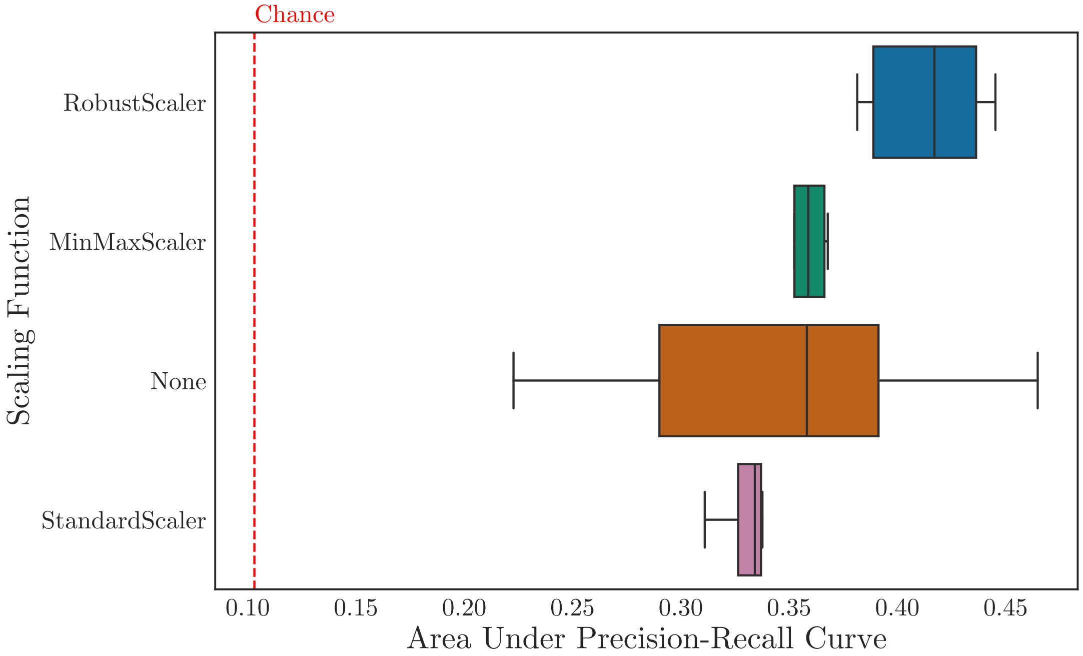


Figure 1.15: Area under ROC for different scaling functions. Scaling functions include: None, StandardScaler (mean: 0, variance: 1), RobustScaler (median: 0, IQR: 1) and MinMaxScaler (min: 0, max: 1). Results presented are aggregated from hyperparameter optimisation performed over entire classification pipeline (including all classifiers).Source: Features (Apr-12) and labels (Apr-14, 2 year forecast window) derived from Master dataset (c. Sep-16).

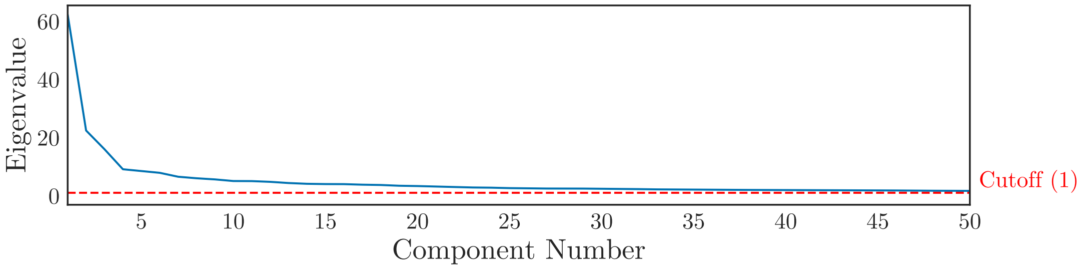


Figure 1.16: Eigenvalues extracted from PCA model. Horizontal line drawn at an Eigenvalue of 1 – this theoretically represents the contribution of one original feature and is commonly used as an approximate threshold for included components. Source: Master dataset (c. Sep-2016).

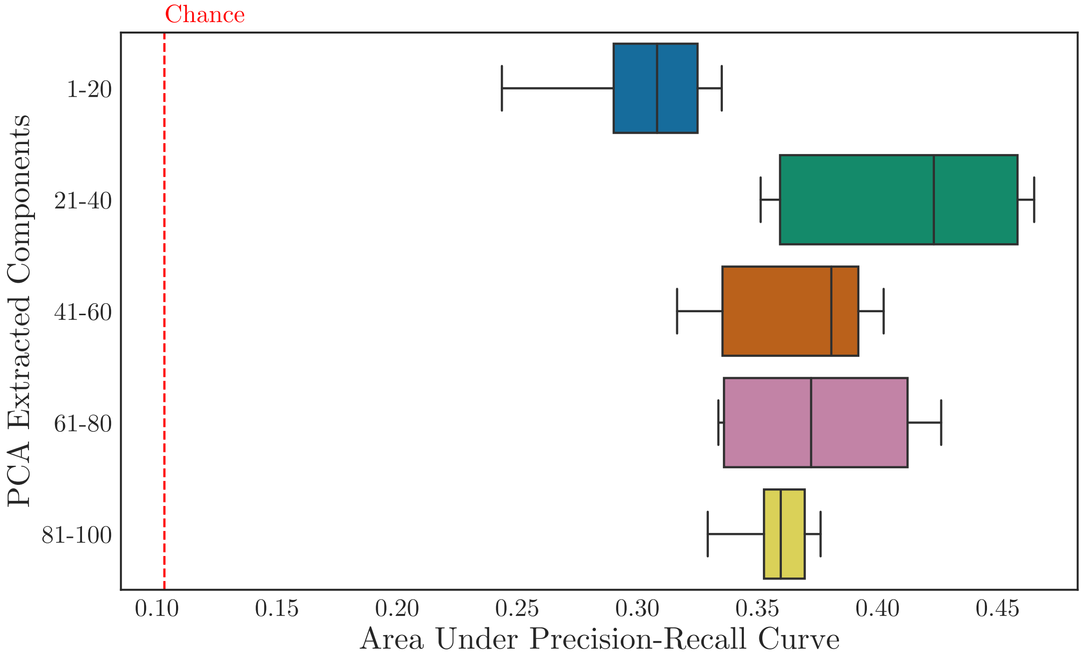


Figure 1.17: Area under ROC for different number of extracted components from PCA. Curves have been grouped by the quotient of the number of components divided by 20 to result in five ordered groups (e.g. Range [0, 19] −¿ 0). Results presented are aggregated from hyperparameter optimisation performed over entire classification pipeline (including all classifiers).Source: Features (Apr-12) and labels (Apr-14, 2 year forecast window) derived from Master dataset (c. Sep-16).

1.2.5 Classification Algorithms

The literature review we performed in the previous chapter revealed seven common supervised classification algorithms potentially suitable for application to this problem area. Our review suggested that Random Forests were most likely to provide a successful trade-off between predictive power, interpretability and time taken. We empirically tested each of these classifiers and compared their performance against a range of metrics, as displayed in Table 1.2. We report maximum as well as median recorded scores to ensure we didn't penalise algorithms that had unfavourable hyperparameter search spaces.

We take a closer look at the Precision-Recall (PR) curves for each classifier in Figure 1.19. While all classifiers perform better than chance, Logistic Regressions and Random Forests come out ahead, and Support Vector Machines and Artificial Neural Networks appear to underperform. Delving into the cross-validated learning curves for each classifier (Figure 1.20) we see that Naive Bayes, Logistic

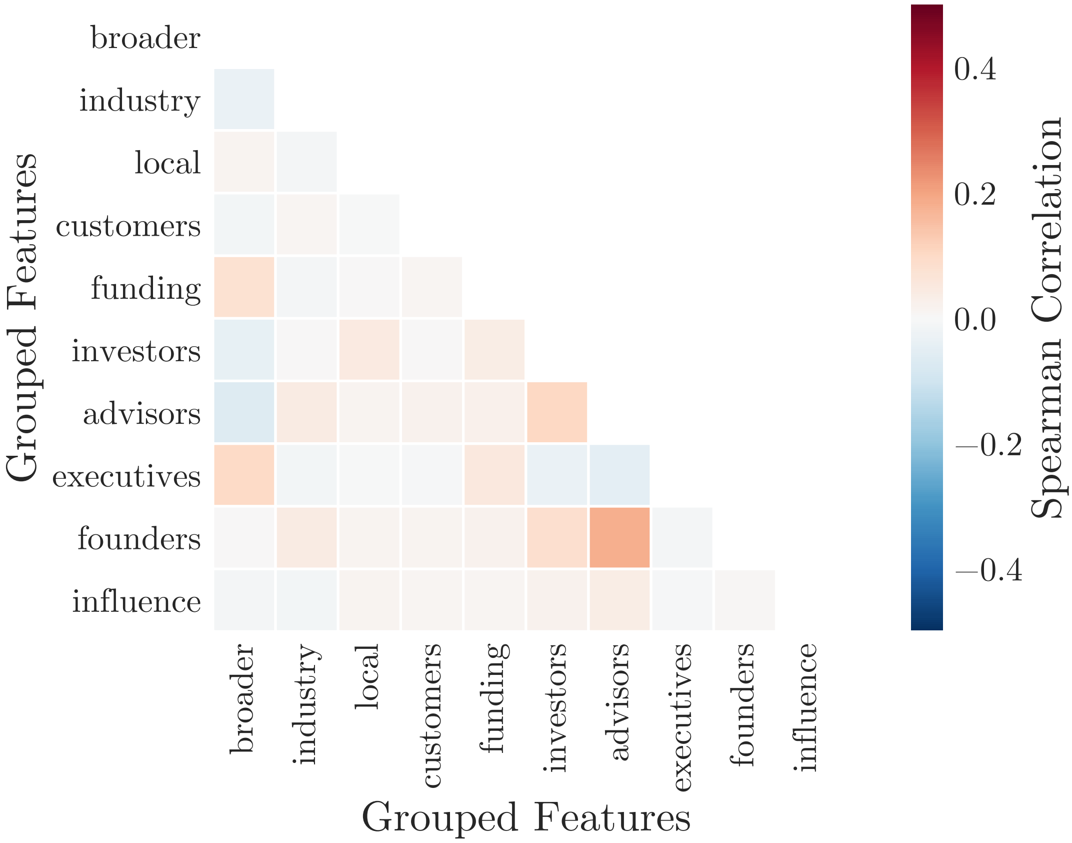


Figure 1.18: Inter-correlations of each factor from conceptual framework. Spearman ranking correlation is used. Individual features are grouped by applying weights that maximise the inter-correlations within each group from our conceptual framework (see Figure 2.1). Source: Master dataset (c. Sep-2016).

	AUC PRC		AUC ROC		F1		Fit Time (s)	
Classifier	Median	Max	Median	Max	Median	Max	Median	IQR
Logistic Regression	0.417	0.465	0.675	0.710	0.339	0.358	7.326	407.031
Random Forest	0.376	0.465	0.619	0.709	0.332	0.360	68.325	29.064
Decision Tree	0.388	0.429	0.651	0.659	0.305	0.314	15.329	6.747
Naive Bayes	0.354	0.367	0.623	0.638	0.303	0.321	8.589	19.131
K-Nearest Neighbors	0.335	0.353	0.532	0.565	0.131	0.226	8.537	12.316
Artificial Neural Network	0.320	0.335	0.517	0.523	0.072	0.096	9.146	12.017
Support Vector Machine	0.233	0.244	0.503	0.504	0.014	0.017	29.015	0.000

Table 1.2

Regression, Artificial Neural Networks and Support Vector Machines quickly converge, whereas Decision Trees, Random Forests and K-Nearest Neighbors require more observations to converge. This suggests that we might expect Random Forests to do better in final testing (as testing will not be cross-validated), as well as in the future as the dataset naturally grows.

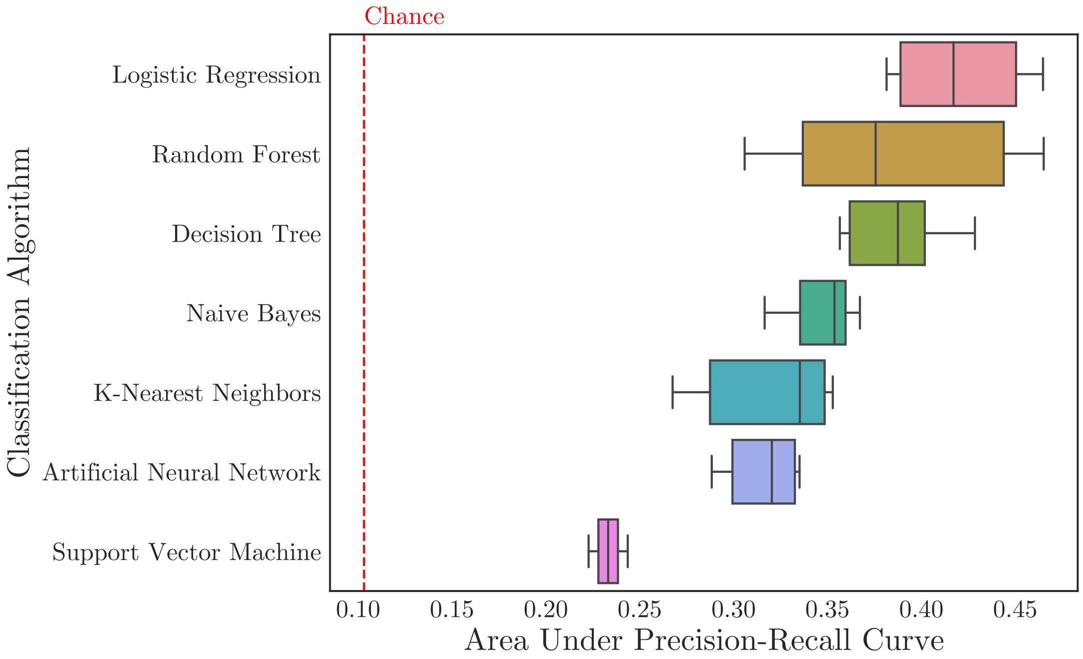


Figure 1.19: Area under ROC for different classification algorithms. All algorithms are implementations from the Sci-kit learn library. Results presented are aggregated from hyperparameter optimisation performed over entire classification pipeline (including all classifiers).Source: Features (Apr-12) and labels (Apr-14, 2 year forecast window) derived from Master dataset (c. Sep-16).

1.3 Pipeline Selection

In the previous chapter, we developed a system that generated a cross-section of candidate pipelines with different hyperparameters. In this step, we terank these candidate pipelines and evaluate the best pipelines (finalist pipelines) over a number of different dataset slices. This process, depicted in Figure 1.21, ensures that our final pipeline is robust in their performance with respect to time. We aggregate the results for each finalist pipeline across these dataset slices and rank the finalist pipelines on their overall performance. Finally, we select the best pipeline.

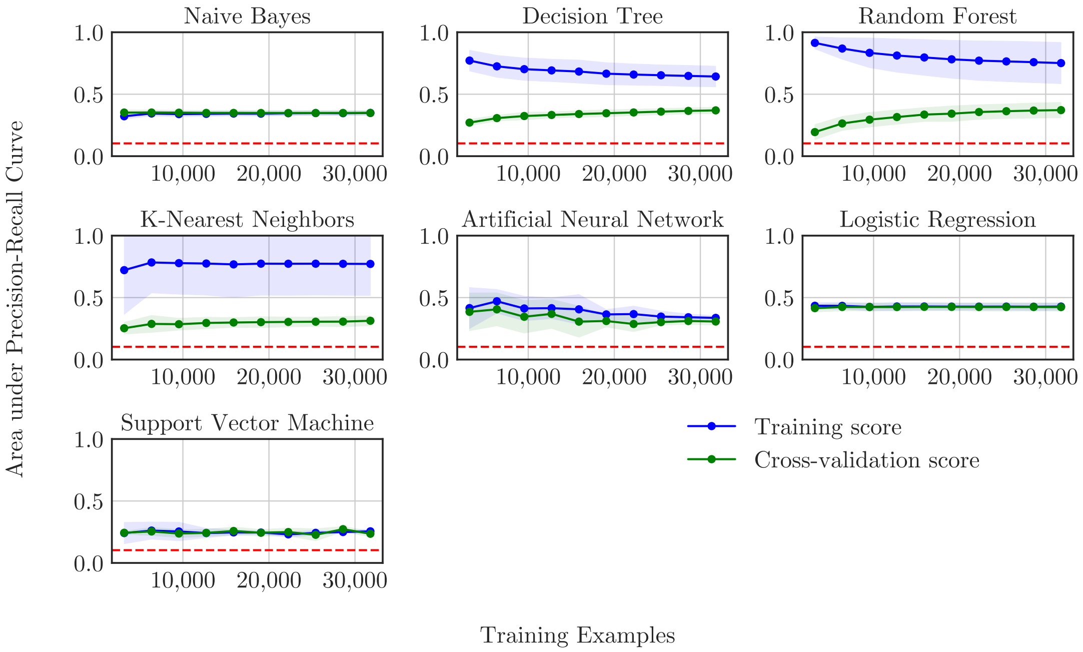


Figure 1.20: Learning curves.

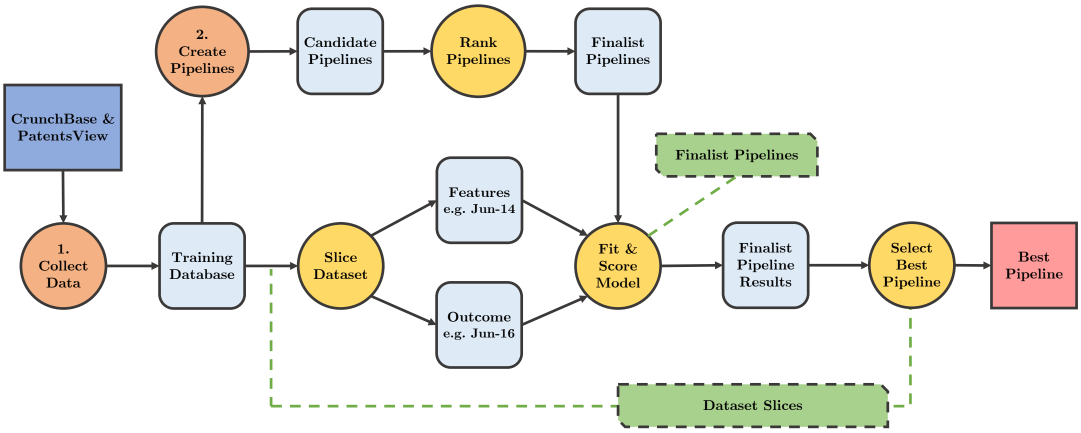


Figure 1.21: Pipeline selection overview.

1.3.1 Dataset Slicing

We developed a procedure for generating historical datasets from our CrunchBase and PatentsView data. CrunchBase provides created and last-updated timestamps for each record in their CSV-formatted dumps (and also in the JSON-formatted responses from their API). We took advantage of this to produce a system that reverse-engineers previous database states by filtering the current database by only records that were created by a given 'slice' date.

We performed preliminary testing of our reverse-engineering technique by comparing a historical CrunchBase database collected in December 2013 with a slice from our primary dataset collected in September 2016, as shown in Figure 1.22. While there are some differences, particularly in the IPO counts, we consider this to be satisfactory variance considering the 3-year time difference (i.e. perhaps some companies have been since removed from the database). The key relations for the purposes of our system are Companies, Funding Rounds and People, all of which had minor differences considering the size of these datasets.

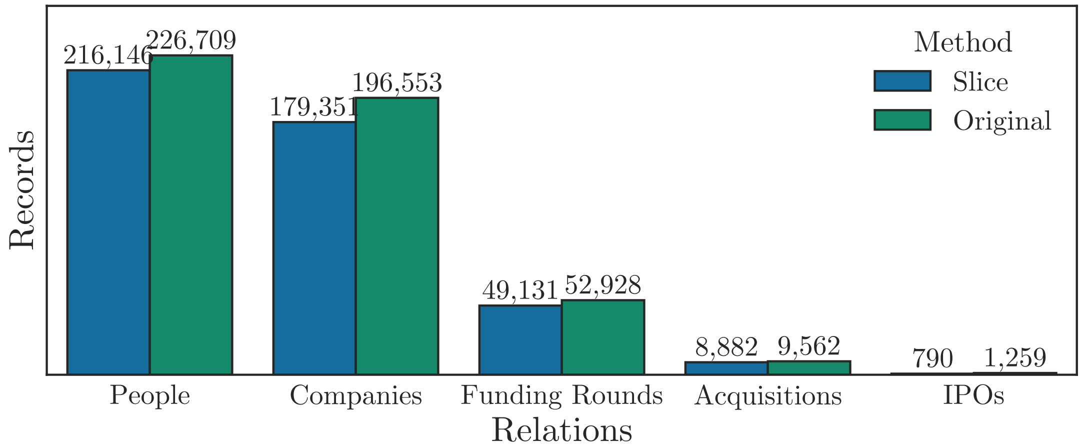


Figure 1.22

Figure 1.23 shows company counts by startup development stage from different dataset slices. We limited our experiments to dataset slices from 2012-onwards because prior to 2012 the datasets become too small to use to make predictions (particularly given the class imbalance).

1.3.2 Evaluation Metrics

Next, we decided how to narrow our candidate pipelines down to finalist pipelines that we can evaluate further. There are a number of different metrics used to evaluate binary classifiers in machine learning. The most simplistic metric is accuracy but this is rarely used in practice because it gives misleading results in the case of imbalanced classes. Receiver Operating Characteristic (ROC) curves are perhaps the mostly commonly used evaluation tool in binary classification, and show how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples. The area under these curves

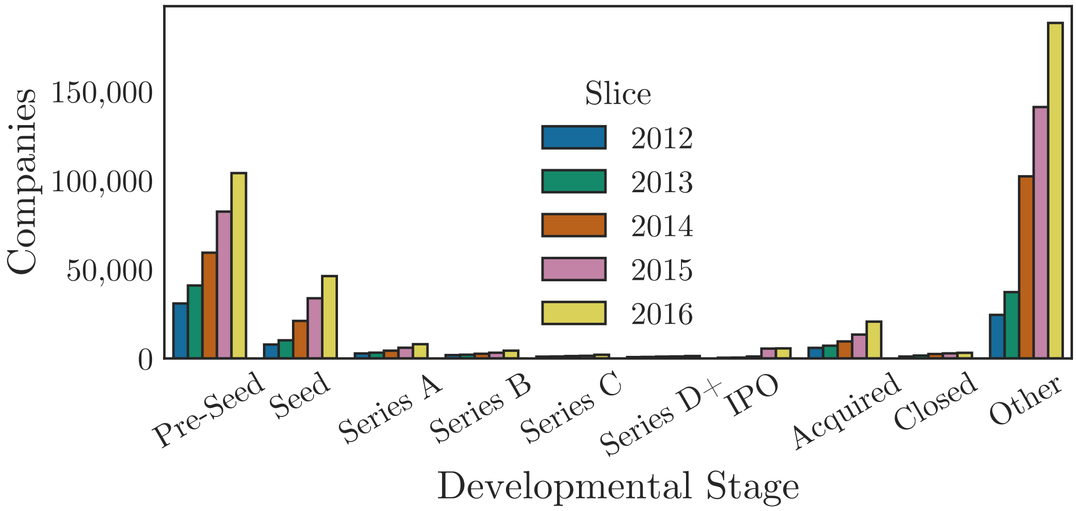


Figure 1.23

gives a standardised result across a spectrum of decision thresholds. Precision-Recall (PR) curves are similar to ROC curves but instead map the trade-offs between precision and recall. They are less commonly used than ROC curves but have been shown to produce more accruate results for imbalanced classes than ROC curves [**Jesse Davis 2006**]. Given our dataset is highly imbalanced (the positive class is approximately 10%) we decided to proceed with PR curves. We will also use this metric to determine which is ultimately the best of our finalist pipelines.

1.3.3 Finalist Pipeline Evaluation

Our hypothesis is that the performance of our classification pipelines may vary with respect to the date that the dataset was collected (in this case, sliced). To study this hypothesis, first we explored variance between the pipelines on aggregate against the slice dates, presented in Figure 1.24. We see little variance on this basis, and we don't observe a relationship between slice date and score.

Next, we study the variance within the individual pipelines, presented in Figure 1.25. Here, we can see there is significantly more variance in the scores. Although there is still a strong positive correlation between the pipelines initial ranking and their scores, we can see that there are some individual deviations. Importantly, the top-ranked pipeline from the first stage actually has a lower median score than the second-ranked pipeline. These results suggest that the top 3-5 pipelines should be evaluated in this manner to ensure that the best

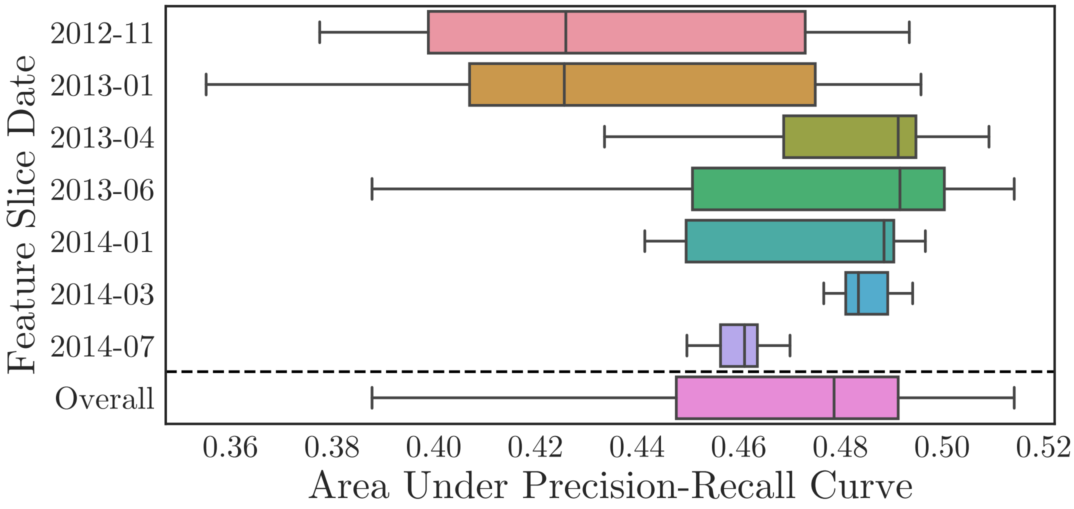


Figure 1.24

pipeline is selected.

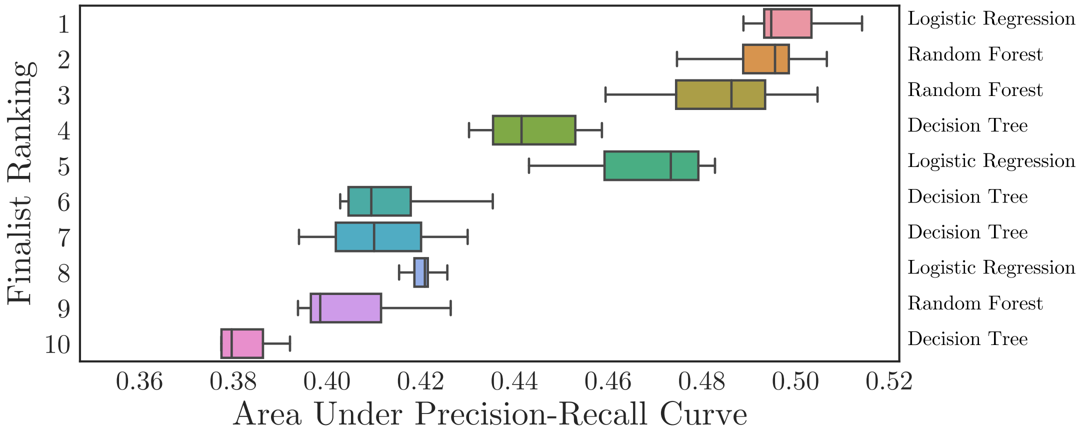


Figure 1.25

The best candidate pipeline is depicted in Table 1.3. We adopted this pipeline configuration for our following experiments.

1.4 Model Fit and Prediction

Finally, we use our optimised pipeline to produce a model and make predictions, as shown in Figure 1.26. Our system applies the best pipeline that was generated

Step	Parameter	Values
Imputer	Strategy	Mode
Transformer	Function	SQRT
Scaler	Function	MinMaxScaler
Extractor	Components	27
Classifier	Algorithm	Random Forest
Classifier	Class Weight	Balanced
Classifier	Criterion	Entropy
Classifier	Boostrap	True
Classifier	Estimators	34
Classifier	Max Depth	8
Classifier	Max Features	SQRT(N_Features)
Classifier	Min Samples Split	2
Classifier	Min Samples Leaf	1
Classifier	Min Weight Leaf	0
Classifier	Max Leaf Nodes	None
Classifier	Min Impurity Split	1e-7

Table 1.3

in the previous section to a training dataset, producing a fitted model. The model is then applied to a feature vector from a held-out test database, which generates a set of predictions which could, in practice, then be used by Venture Capital (VC) firms. We evaluate the accuracy of the models produced by our system with respect to a number of variables in the next chapter, Chapter **??**.

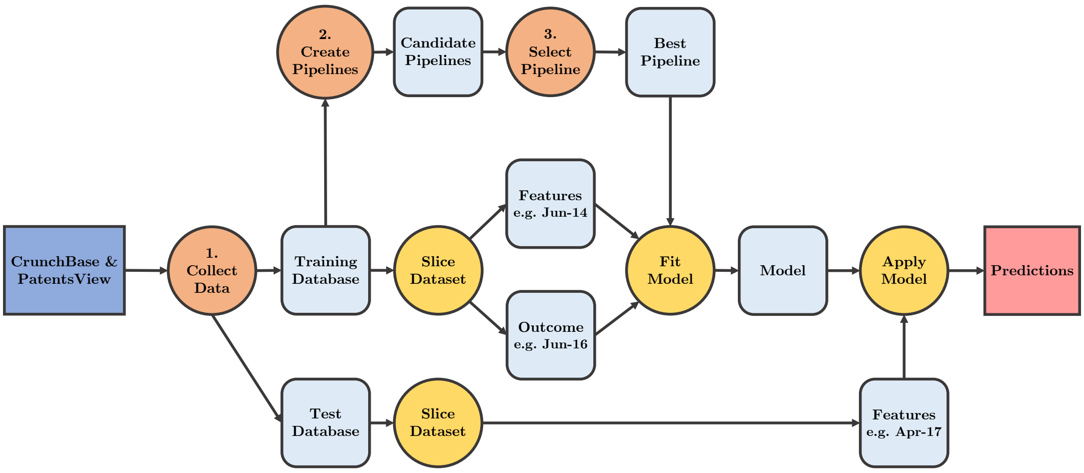


Figure 1.26: Model fit and prediction overview.

Bibliography

[1] Schultz, L. A. “Preliminary Patent Searches: New and Improved Tools for Mining the Sea of Information”. In: *Colo. Law.* 45 (2016), p. 55.