Predicting Patent Success: A Machine Learning Approach to Commercialization Prospects

Valerio Ardizio, Stefano Viel, Malena Mendilaharzu
Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—Anticipating the commercial success of patents holds significant importance for research institutions, inventors, and industry stakeholders. This paper investigates the convergence of machine learning techniques and patent commercialization to develop a tool able to predict the likelihood of a patent's success. In the pursuit of this objective, we offer a thorough examination of the framework employed as well as valuable insights into the key information required for accurate predictions. Using this methodology allowed us to achieve an accuracy of 0.927 and an F1-score of 0.927.**

I. Introduction

Our study aims to construct a predictive model capable of assessing the likelihood of a patent being commercialized. This involves analyzing various factors, including the type and location of the industry, as well as the citation patterns associated with the patent, among others. This report offers an outline of the methodology employed to create such a classifier. Section II presents the IPRoduct dataset [1] and the exploration of different preprocessing techniques. Section III explains the selection of our model while section IV delves into the intricacies of its training and fine-tuning of its hyperparameters. Subsequently, section V assesses the hierarchical importance of each feature in predicting the likelihood of a patent's success. Finally, section VI offers the performance and outcomes, and section VII presents a concise summary of the work conducted. Section VIII was included to discuss the imperative need for ethical considerations tied to the deployment of machine learning in predicting patent success.

II. Data Analaysis and Preprocessing

Our investigation relies on the IPRoduct dataset, initially developed at EPFL for a virtual patent marking initiative to provide a link between products and the patents that protect them.

The IPRoduct dataset is composed of 63349 samples, each encompassing 772 distinct features. It consists of a relatively balanced dataset with approximately 55% of the samples falling into class 0 (non-commercialized) and the remaining 45% falling into class 1 (commercialized).

Upon inspecting the dataset, a couple of challenges emerge:

1) Firstly, a substantial proportion of the dataset exhibits missing values, accounting for roughly 38% of the total entries.
2) Secondly, out of the 772 distinct features, 765 consist of numerical ones (including Google embeddings[1]) while the rest consist of textual features. Consequently, addressing the mixed-type nature of the dataset becomes essential in the creation of our framework.

A. Dealing with missing values

In addressing the first concern (1), we explored the possibility of dropping or substituting the missing values with either zeros or statistical measures of their corresponding features, including the mean, median and the most frequent value. Following a series of experiments, we observed that all four techniques yielded very similar performances (refer to Table I).

It must be specified that the following results were derived using logistic regression as the baseline model. To enhance result reliability, we employed a 5-fold cross-validation, and each replacement method was independently applied to the training and validation sets to prevent biased outcomes.

Method	F1 Score	Accuracy	Precision	Recall
Mean	0.8769	0.8929	0.8894	0.8647
Median	0.8772	0.8932	0.8897	0.8650
Most Frequent	0.8772	0.8932	0.8898	0.8649
Zero	0.8770	0.8931	0.8896	0.8648

TABLE I: Average Performance Metrics for Different Imputation Methods. Logistic regression with 1000 as max iterations is used for the testing.

Moreover, we conducted ANOVA[2] tests to assess the variations in accuracies across each method. The analysis of variance (ANOVA) revealed an F-statistic of 0.186, and an associated p-value of $p = 0.904$. Given the non-significant p-value, we fail to reject the null hypothesis, suggesting no substantial difference in the accuracies among the groups. Hence, our selection of the most frequent replacement was based on its slightly superior performance, but we acknowledge that the observed differences are not considered significant.

B. Dealing with a combination of text and numerical features

In response to (2), we needed to design a strategy to process text and numerical features differently with the goal of integrating them into a unified model later on.

[1]Google embeddings
[2]We used the SciPy library to perform statistical tests

For what concerns the numerical features, a particular pre-processing was necessary as we aimed to test the normalization or scaling of these features to ensure uniformity and to prevent the model from assigning undue importance to those with larger magnitude. Meanwhile, for textual features, some could be easily converted into a numerical format, requiring straightforward one-hot encoding. This applied to features such as the list of countries where the patent was published and the company responsible for its publication. However, more intricate cases, such as features comprising entire paragraphs, demanded alternative techniques. Specifically for the abstract and the text description, even though the dataset already included an embedding of the full patent text (namely, the Google embeddings that are identified by all variables starting with `ge`), we tried to design our personal strategy to process such features in the attempt to extract further useful information. In this context, we explored three primary approaches:

1) Count Vectorizer[3]: representing our simplest approach, consists of creating a very sparse matrix where each row corresponds to a document, and each column corresponds to a unique word. The entries consist of the count of the occurrences of each unique word.
2) Word2Vec[4]: consists of a neural network-based model that learns the distributed representation of words in a continuous vector space. It represents words in such a way that the ones semantically similar are close to each other in the vector space. This should facilitate the classification procedure as words that are similar in meaning will have similar vector representation [2].
3) TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer[5]: evaluates the importance of a word by assigning them weights based on their frequency and rarity on the entire document.

The two textual features that required vectorization were the abstract and the description text, consisting of the textual description of the images in the patent. The latter is much longer than the first so due to computational restrictions we only considered the abstract in the following comparison.

As in the previous section, the comparison of the different vectorization methods was conducted through 5-fold cross-validation (refer to Table II).

Metric	**F1 Score**	**Accuracy**	**Precision**	**Recall**
Baseline	0.8772	0.8932	0.8898	0.8649
Count vectorizer	0.8740	0.8907	0.8869	0.8615
Word2Vec	**0.8744**	**0.8911**	**0.8875**	**0.8618**
TF-IDF Vectorizer	0.8734	0.8901	0.8858	0.8614

TABLE II: Average Performance Metrics for Different Vectorization Methods. Logistic regression with 1000 as max iterations is used for the testing.

Our investigation revealed that among these three techniques, Word2Vec leads to the best results. However, we came to notice that utilizing the word embeddings for the abstract did not improve the accuracy of our baseline model, as can be deduced from Table II. Hence, we decided to not use them, as they did not seem to provide with valuable information for the prediction of a patent's success.

III. Model Selection

Subsequently, we performed an analysis of the performances of the most widely used models in the literature to asses which were the most appropriate for our task. This is a list of the models we took into consideration:

1) Random Forest
2) Logistic Regression (max iterations = 1000)
3) Neural Network (max iterations = 1000)
4) Gradient Boosted Trees
5) Support Vector Machine (SVM)
6) Word embedding for textual features using BERT + Neural Network

All the above-cited models where taken from the scikit-learn[6] library and were trained using the default parameters.

These models were chosen to capture a spectrum of techniques, from ensemble methods like Random Forest to deep learning techniques such as Neural Networks. They were all trained and tested only on the numerical features as the previous section showed no improvement from adding the vectorized version of the abstract.

The only model for which we used textual features is BERT (Bidirectional Encoder Representations from Transformers) [3]. We downloaded a pre-trained version of the BERT[7] tokenizer and model.

BERT is a pre-trained natural language processing model that utilizes transformer architecture. Developed by Google, BERT is designed to understand the context and nuances of words in a sentence bidirectionally, allowing it to capture complex linguistic relationships and improve performance in various language-related tasks. BERT incorporates self-attention [4] as a key mechanism within its transformer architecture. Self-attention allows the model to evaluate the significance of each word in a sequence relative to others, enabling comprehensive consideration of contextual dependencies. All these features make it substantially different from the vectorization techniques seen previously, that's why we only consider it in this section.

The pre-trained version of BERT and the implementation of the additional layer were done through the PyTorch and Transformers library[8]. We have only trained the last layer of BERT, while all the others were left frozen. The output of the model, which represented the low-dimensional version of the abstract, was combined with the features and fed into two fully connected layers with 128 hidden units. The employed optimizer was AdamW [5] and cross-entropy was used as the

[3]Implemented by utlizing the scikit-learn library

[4]Implemented by utilizing both the NLTK library and the gensim library

[5]Implemented by utlizing the scikit-learn library

[6]scikit-learn library

[7]The BERT model and tokenizer used in this project are obtained from Hugging Face (https://huggingface.co/bert-base-uncased) specifically the employed version of BERT is bert-base-uncased.

[8]PyTorch library, Tranformers library

loss function. Training was performed for 12 epochs and with a batch size of 32. The training loss is reported in Fig. 1

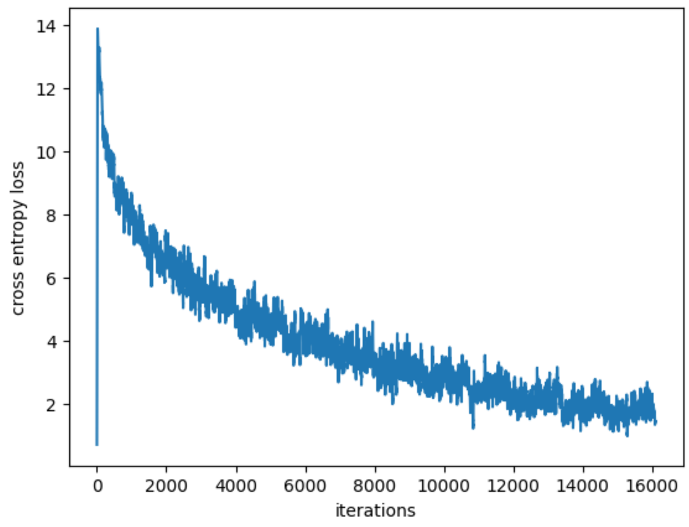

Fig. 1: Training loss for BERT

Table III details the performance metrics, including F1 score, accuracy, precision, and recall values for each one of the aforementioned frameworks. Based on this information, we selected model (1), Random Forest, as our optimal one. It must be noted that BERT was trained both with and without the Google embeddings (abbreviated as g.e.) and the results were included in Table III for both cases.

Model	F1 Score	Accuracy	Precision	Recall
Random Forest	**0.911**	**0.922**	**0.917**	**0.904**
Logistic Regression	0.878	0.894	0.893	0.863
Neural Network	0.895	0.907	0.892	0.900
Gradient Boosted Trees	0.885	0.901	0.904	0.867
SVM	0.882	0.898	0.906	0.859
BERT (all features)	0.906	0.905	0.905	0.0904
BERT (without g.e.)	0.903	0.901	0.904	0.0902

TABLE III: Average Performance Metrics for Different Models

Since we were able to reject the null hypothesis with ANOVA, we utilized the T-tests[9] to quantify the pairwise statistical significance of differences between the accuracies obtained through the models taken in consideration. This approach enabled us to confirm that random forest was statistically significantly better than all the other models.

IV. Random Forest: Training and hyperparameter tuning

To train our model effectively, we employed a 5-fold cross-validation in combination with an exhaustive grid search across various hyperparameters values.

[9]We used the SciPy library to perform statistical tests

The random forest algorithm is an ensemble technique that constructs a multitude of decision trees during training and provides output by aggregating their predictions. Employing cross-validations enhances the model robustness by partitioning the dataset into k folds, training on $k-1$ of them, and validating on the remaining one, allowing to use the whole dataset for both training and testing.

Through an exhaustive grid search, we explored diverse values for our four hyperparameters. What follows is a list of such hyperparameters and the values that were taken into consideration:

- `n_estimators`: the total number of decision trees. `n_estimators` = $[50, 100, 200]$
- `max_depth`: the maximum depth of each tree. If None, then nodes are expanded until all leaves are pure. `max_depth` = $[None, 10, 20, 30]$
- `min_samples_split`: the minimum number of samples required to split an internal node. `min_samples_split` $[2, 5, 10]$
- `min_samples_leaf`: the minimum number of samples required to be in a leaf node. `min_samples_leaf` = $[1, 2, 4]$

Fine-tuning `n_estimators`, `max_depth`, `min_samples_split` and `min_samples_leaf` involves a trade-off between model complexity, generalization performance, and computational efficiency.

On one hand, the more trees (`n_estimators`) our forest has, the more robust our model becomes against overfitting but the higher the computational costs are. On the other hand, the deeper a tree is (`max_depth`), thee higher the capacity to capture patterns but the more likely our model is to overfit.

Further, `min_samples_split`, `min_samples_leaf` play a key role in controlling the structure of the decision trees within the ensemble. A higher `min_samples_split` prevents the model from creating nodes with a small number of samples, making the decision tree less prone to capturing noise in the training data. Similarly, a higher `min_samples_leaf` value, enforces a minimum size for leaf nodes, preventing the model from creating overly specific structures. Results are reported in Section VI.

V. Searching for more relevant features

An important objective of our project was to establish the hierarchical importance of features for predicting the likelihood of patent commercialization. We proceeded to use a random forest to obtain each of the feature's importance scores. More specifically, the contribution of a feature in making predictions was done by evaluating the Gini impurity reduction when that particular feature was used to make the split (refer to Fig. 2).

It is interesting to notice that `vpm_patent_score` is by far the most important feature. This represents the ratio of patents present on the Virtual Patent Marketing platform to the total number of patents owned by the company. The company from which the patent comes has great importance as well as the number of backward and forward citations. Interestingly the Google embeddings for the patents play a significant role, suggesting that the text of the patent can also be reliably used to predict commercialization. Therefore, the patent text has significant information and we were not able to extract it in Section V. This could be due to the fact that we were restricting our feature extraction methods only to the abstract and were only deploying basic vectorization techniques.

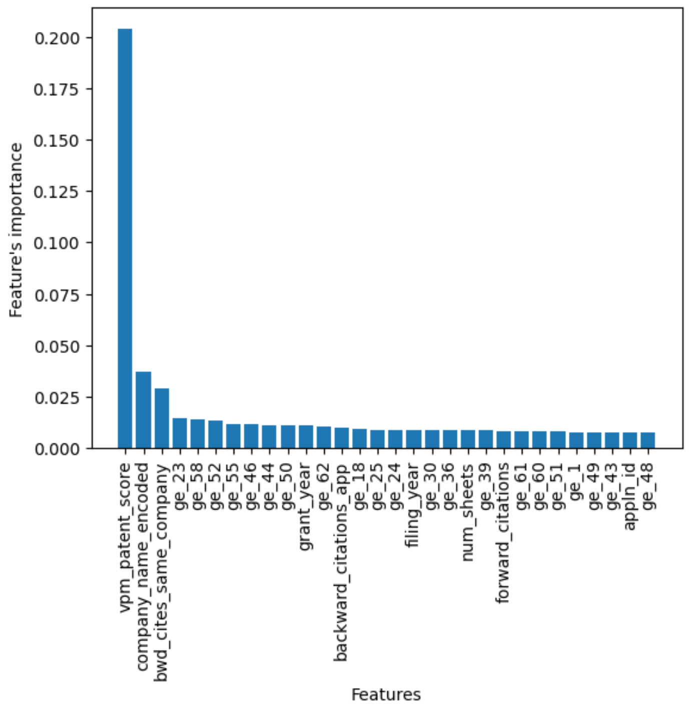

Fig. 2: Feature importance scores

After conducting an initial exploration of the features, we proceeded with more refined experiments on subsets of our processed dataset. The objective was to assess how removal of specific features can impact the performance of our classifier. The following is a list of groups of features which were considered.

1) All features (baseline);
2) Removing the Google embeddings;
3) Removing all features containing future information, such as future patent citations, as these values are unknown;
4) Removing the virtual patent marking score;
5) Applying Principal Component Analysis with 627 components (value calculated to retain 95% of the variance in the original data);
6) Implementing Chi-square feature selection, evaluating feature independence from the target variable to select the most informative ones;
7) Removing the Virtual Patent Marking score, and the futures information.

Table IV presents the outcomes of these experiments, following an evaluation of the accuracy, F1 score, precision, and recall values.

Case	F1 score	Accuracy	Precision	Recall
Include all columns	0.913	0.924	0.921	0.905
No google embeddings	0.900	0.914	0.916	0.885
No future information	**0.917**	**0.927**	**0.924**	**0.909**
No VPM patent score	0.883	0.897	0.894	0.872
PCA with 627 components	0.865	0.882	0.891	0.840
Chi-square feature selection	0.915	0.927	0.922	0.908
No future information & VPM	0.875	0.891	0.887	0.863

TABLE IV: Performance Metrics for Different Subsets

It is interesting to notice that removing future information doesn't lead to a reduction in the model performances. Therefore, we can expect to reach the same accuracy at the moment of patent publication. As expected, the most significant drop in performance where found when removing the VPM patent score.

VI. Results

The results presented in Table V were obtained after applying the techniques outlined in sections II to V.

The optimal model was Random forest with the following hyperparameters:

1) `n_estimators` = 200
2) `max_depth` = *None*.
3) `min_samples_split` = 2.
4) `min_samples_leaf` = 1.

Employing these values yielded the following results:

	F1_score	Accuracy	Precision	Recall
Tuned hyperparameters	0.927	0.927	0.926	0.925

TABLE V: Optimal Results

VII. Summary

In this project, we explored various machine learning models to forecast the likelihood of a patent's commercialization, considering factors like industry type, location, and citation patterns. Our focus was on identifying key features that could explain whether a patent would be commercialized or not. Unfortunately, the additional methods that we deployed for the inclusion of textual features, such as the abstract, proved almost negligible in enhancing our model's performance. We anticipate that this research sheds some light into the intricacies of patent commercialization as well as potentially serving as a foundation for a more informed analysis of the factors influencing a patent's success.

To improve our model's performance, we aim to investigate the use of text description features for word embedding, assessing their potential utility compared to the abstract. However, due to the limited scope of this project and of our computational resources, we deferred this exploration to future endeavors.

VIII. Ethical Considerations

Within the present section we aim at enlightening some aspects regarding concerns that should be examined in order to ensure the ethical integrity and societal implications of our predictive model.

A pertinent ethical risk that can be found within our work regards the potential bias associated with the company filing the patent. After observing a strong discrepancy in the commercialization success rates between Nokia, for which 99.9% of the patents filed were commercialized, and BlackBerry, for which the percentage of commercialized patents is only 11.4%, we were clearly able to understand that there was a potential bias at stake, raising concerns about the impartiality of our model.

The primary stakeholders affected by such a bias can be identified both by certain companies, such as BlackBerry, that have had a historically lower success rate within the realm of patent commercialization, as well as the researchers working for such enterprises, who will find themselves limited in the amount of recognition one of their projects can potentially receive. The negative impact of our model lies on the reinforcement of the existing disparities in the commercialization rates, potentially obstructing having fair opportunities in the field of innovations.

In order to assess this risk, we had to evaluate its severity and its likelihood. On one hand, the severity is high in a concerning way, since biased predictions can influence decision-making processes, potentially favoring companies with higher historical success rates. On the other hand, the likelihood of occurrence of such a scenario is considerable as well, given the observable disparity in success rates among different companies, even concerning the same field of interest, within the dataset.

Our approach to enhance the fairness of our predictive model relied on dropping the feature regarding the name of the company filing the patent. However, the precision of our predictions decreased, highlighting, as it was expected, an unavoidable trade-off between ethical fairness and high performance of our model. As the original dataset was not originally created to predict patents' success, we suggest a thorough examination of its potential biases before utilizing the tool developed in this paper. Further, given the limited scope of this project, both collecting additional data and creating artificial data appeared to be unfeasible, but could be beneficial for future enhancements.

Acknowledgements

The authors thank Prakhar Gupta for the guidance and support he provided during this project.

References

[1] G. de Rassenfosse and K. Higham, "Wanted: a standard for virtual patent marking," *Journal of Intellectual Property Law & Practice*, vol. 15, no. 7, pp. 544–553, 06 2020. [Online]. Available: https://doi.org/10.1093/jiplp/jpaa063

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://doi.org/10.48550/arXiv.1301.3781

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1810.04805

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.1706.03762

[5] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1711.05101