Final Report – Team 94 Developer Salary Prediction

# Introduction

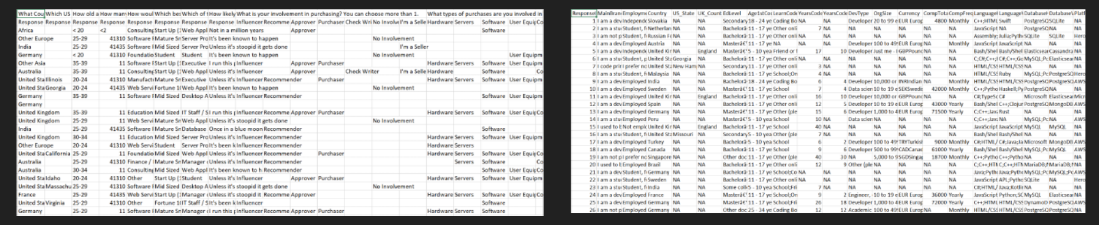
As one of the fastest growing sectors, the Tech Industry has positioned itself as a lucrative field that offers rewarding benefits and flexibility that makes it highly attractive compared to its counterparts. The fast pace also means, it is ever evolving with rapid technological advancements which require continuous improvement and upgrading of skills for a successful career.

## Problem Statement

In this project, we want to understand the requirements for the different developer roles, and the salary projections for the various roles. We used the Developer Survey dataset from Stack overflow to determine the demand of developers within future years and identify the skills that will come at a premium rate.

## Data Description and Background

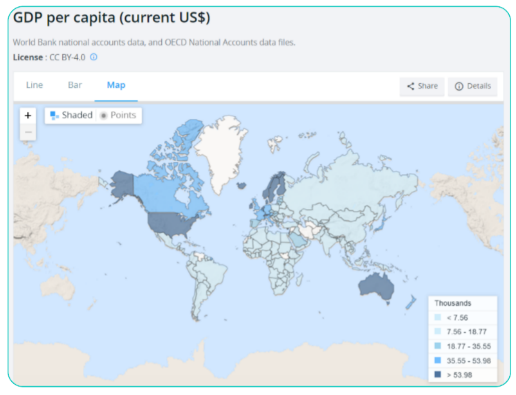
Our data is based on the annual Stack Overflow Developer survey. An annual survey of programmers, software development professionals, and others associated with the computer science industry. All survey response answers are available as datasets for each year from 2011 to 2021. We extracted a rich analytical dataset for each individual year, as well as a consolidated analytical dataset that contains comparable answers across each year. For the consolidated dataset gathered information on respondents: country of residence, occupation, company size, years of IT experience, programming language competencies, job satisfaction and compensation/salary.

*Preview of unnormalized survey data:*

## Supplementary Data

In order to enrich our data analysis, the team incorporated GDP per capita data. We incorporated a normalised version of the GDP data across all years to be used in the normalised, consolidated dataset. The goal is to use a normalised version of GDP per capita in the dataset across all years as an additional attribute to determine if this a factor which influences developer salaries.

*Preview of GDP per capita data, visuals provided by source website.*



## Hypotheses

Our key hypothesis is that it is possible to make reasonable predictions of an individual's current salary and forecast future potential salary from a set of basic explanatory variables related to the survey data. Supporting hypotheses include, firstly, there are certain programming languages and knowledge of technologies that will have a meaningful impact on salaries. The impact of these will differ depending on country of residence, years of experience, current occupation, and vary across the years this survey has been administered. Secondly, we also hypothesize if there was a salary premium of working at the office prior to the pandemic, this has diminished during and after the pandemic. These were our initial hypotheses, however due to the time constraints, we focussed primarily on the which attributes contributed most to the salary classification of a developer.

# Planned Approach

Our planned approach is as follows:

1. Initial Exploratory Data Analysis
2. Methodology
3. Data Transformations
4. Analyse Results

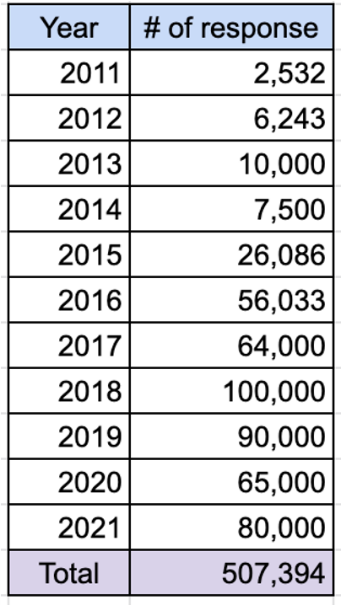
This was the initial planned approach; however, we have realised when working with survey data the amount of normalisation and data cleansing activities required across the datasets can be cumbersome. Below is the final timeline of our project:



## Data Cleaning Progress

Extraction of attributes from survey questions from 2011 to 2021 was challenging, as every year questions were asked differently, and options provided for those questions were also not same. For example, in 2012 there were around 10 programming languages to choose from for the respondents whereas in 2018 there where more than 15 languages. One way to tackle this problem is to have a most common list of attributes which are answered across every year.

Utilising the [Python script](http://ttps://github.gatech.edu/MGT-6203-Summer-2022-Canvas/Team-94/blob/main/salary_prediction_for_software_developers/Scripts/download_survey_data.py), we downloaded the row survey questions and responses for 11 years.



*Year wise number of responses*

We performed following 4 steps to clean the data.

**Step 1: Mapped survey questions to listed initial key attributes for every year**. Notebooks in this Git [folder](https://github.gatech.edu/MGT-6203-Summer-2022-Canvas/Team-94/tree/main/salary_prediction_for_software_developers/Notebooks/data-extraction) has the detailed mapping of every question to key attributes. There are a lot many attributes beside these attributes, we considered these attributes could factor the salary prediction based on the previous search done on this field.

*Initial Attribute List*

*Country*

*Age*

*IT\_experience\_in\_years*

*Company\_size*

*Occupation*

*Desktop\_OS*

*Proficient\_languages*

*Compensation*

*Gender*

*Industry*

*Current\_development\_project*

*Product\_technology*

*Stackoverflow\_reputation*

**Step 2: Merge the data across all years.**

Not all the attributes were there across the years. For merging some attributes were marked as unknown and some are imputed. More details can be found in the following [notebook.](https://github.gatech.edu/MGT-6203-Summer-2022-Canvas/Team-94/blob/main/salary_prediction_for_software_developers/Notebooks/data-extraction/data-consolidator.ipynb)

**Attribute merge validation:**

* **Country**

All the years has country information will be used for merging as is. This attribute is used to fetch the GDP per capita and that attributes is used for model development.

* **Age**

2016 has both age rage and specific age value, since most of the years has rage retaining the range as the values.

2017 has no age column will be imputed as number of years of experience+22 as 22 is the minimum age one attains after degree.

* **IT\_experience\_in\_years**

2016 has both experience range and specific values, since most of the year's gas range, retaining the range the values.

* **Industry**

Industry attributes is available only for 2011, 2012, 2013, 2014, 2015 and 2016 (6 years) and missing in 5 years which would be hard to impute, hence dropping this column.

* **Company\_size**

2015 needs to be imputed for now all the values will be marked as unknown, later was mode imputed.

* **Occupation**

All the years has occupation information will be used for merging as is.

* **Current\_development\_project**

This attribute is available only in 2011 and 2012 and their values are not significant for our analysis, hence dropping this attribute.

* **Proficient\_languages**

This attribute has list of languages for all the years hence will be used as is for merging. In further cleaning process only first, proficient language will be considered for modelling.

* **Desktop\_OS**

2017 desktop information is captured as part of platform worked on.

* **Job\_satisfaction**

This attribute is available only for 7 years for the years where this attribute is not available will be marked as unknown.

* **Gender**

This attribute is missing for 2011, 2012 and 2013. Values are imputed based on industry standards for each year separately.

* **Compensation**

2016 there is range and mid value for salary, would be considering range as model output is range.

**Step 3: Normalising individual attributes.**

Attributes responses for every year was in different range, these attributes are normalized individually, and missing attributes were imputed by mean/mode. This [folder](https://github.gatech.edu/MGT-6203-Summer-2022-Canvas/Team-94/tree/main/salary_prediction_for_software_developers/Notebooks/data-normalization) contains notebooks for individual attributes and their normalization process.

**Step 4: Encoding for normalized data for model development.**

All the categorical attributes are one hot encoded using sklearn`s make column transformer. This [notebook](https://github.gatech.edu/MGT-6203-Summer-2022-Canvas/Team-94/blob/main/salary_prediction_for_software_developers/Notebooks/model_data_encoder.ipynb) has detailed code for encoding of attributes. With these 4 steps we were able to normalise the attributes across the 11 years of survey data. Data size was reduced from 507 thousand rows to 250 thousand rows with 60 encoded attributes. Consolidated list of attributes across 11 years of survey data are shown below.

*Final attribute list:*

*GDP*

*Age*

*IT\_experience\_in\_years*

*Company\_size*

*Proficient\_languages*

*Desktop\_OS*

*Job\_satisfaction*

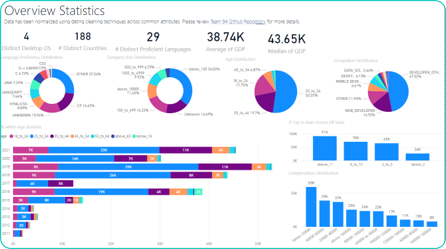
*Gender*

*Compensation*

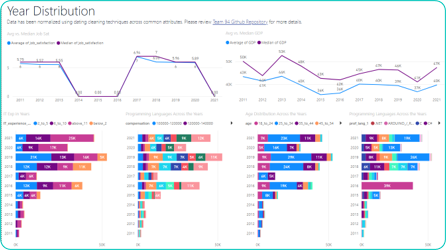
## Exploratory Data Analysis

Once we had the consolidated and normalised dataset, we used Microsoft Power BI to create visuals across the years of survey data for exploration and find key insights or interesting anomalies.

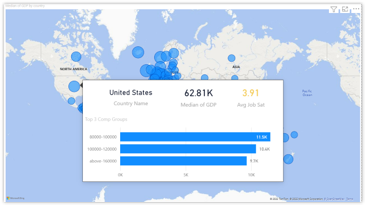
We created three views in the report; Overview Statistics, Year Distribution, and GDP Map Distribution. The Overview Statistics report provides a holistic overview of the data across all years, with a breakdown of programming languages, age brackets, occupation distribution, company size distribution, IT experience brackets and Compensation distribution.



Year Distribution is a report which provides a breakdown of the key attributes of interest across every year, for instance, job satisfaction from 2011 to 2021, and GDP average vs GDP median from 2011 to 2021. Other visuals of interest on this view are the breakdown of IT experience in years, Compensation, Age, and Proficient Coding Language all broken down by years.

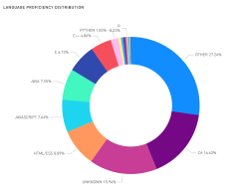
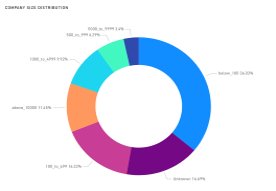
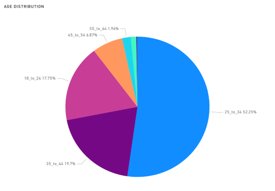


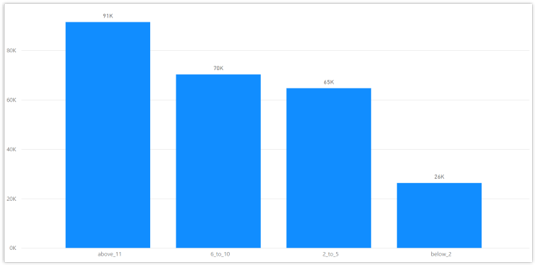
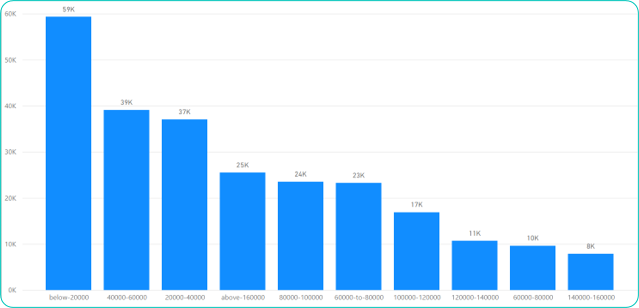
Finally, the last view is a representation of GDP per capita around the world with a drill-through insight of top 3 Compensation buckets, median GDP and average Job Satisfaction within each country.

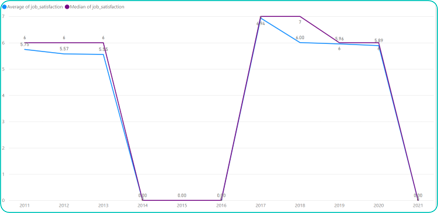
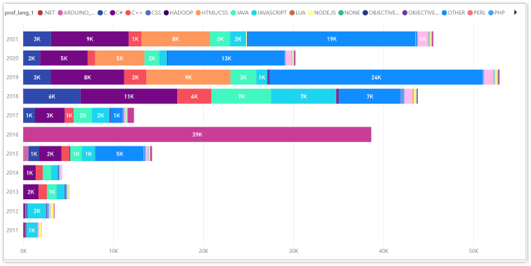


### Key Insights – EDA

We notice some key metrics within the dataset, there are 4 distinct operating systems, 188 distinct countries, 29 distinct programming languages, and 38.74K Average GDP and 43.65K Median GDP. Notice that most of the age distribution is in the 25-34 age range at 52%. Language proficiency is comprised of 27% denoted as “Other” while there is also a substantial amount of Unknown in this data, about 15% of the data. The first proficient language denoted as the next highest is C# comprising ~17% of the data. The majority of company size distribution is the combination of below 100 employees and 100-499 employees, which is roughly 50% of the data collectively. Most of the salary compensation is denoted by the bucket below 20K, and IT experience majority is over 11 years of experience. We notice some interesting Unknown, or Null data behaviour in Proficient Language, this question was not asked in the 2014 survey, so this was denoted as Unknown for the entire year results in the normalised data. We also see Job Satisfaction as 0 for years 2014-2016 as this question was not available during these years of the survey. Visuals of these findings can be found below.







# Modelling

## Overview of approach and model evaluation

Our modelling strategy was based on ensemble models including light gradient boosting machine (LGBM) and random forests, as well as multinomial logistic regression. We selected ensemble models considering their potential to improve performance over a single estimator due to reduced variance, and logistic regression was used for comparison which, unlike its linear counterpart, does not assume linearity or homoscedasticity. In this supervised learning task, salary is an ordinal variable – to explore how the violation of assumptions about ordinal data type impacts model performance, for each model category (boosting, bagging, and regression) the outcome variable was treated as continuous or categorical. Modelling was implemented in Python 3 using libraries including Scikit-learn and LightGBM.

To use salary values as a continuous data type, each of the nine salary ranges were mapped onto integer values (1 to 9) prior to model fitting. For reference, the following mapping of integer values to salary levels was used:

* 1: below-20000.
* 2: 20000-40000.
* 3: 40000-60000.
* 4: 60000-80000.
* 5: 80000-100000.
* 6: 100000-120000.
* 7: 120000-140000.
* 8: 140000-160000.
* 9: above-160000.

As the models return floats, two strategies for rounding to integer salary levels were investigated. These were based on nearest-integer rounding and an optimization function using the Nelder-Mead algorithm implemented in order to find level thresholds to maximize the κ metric (see below) for a given continuous output.

To evaluate the similarity between true values and model predictions, the quadratic weighted κ metric was used. The metric is constructed as outlined in the evaluation [section](https://www.kaggle.com/competitions/petfinder-adoption-prediction/overview/evaluation) of a Kaggle competition (‘PetFinder.my Adoption Prediction’). Briefly, a square matrix is constructed such that corresponds to the number of records that have salary value of (actual) and received a predicted rating . Another square matrix of weights, , is calculated based on the difference between the actual and predicted values:

Where the number of data points. An - by- matrix of expected ratings is created, computed as the outer product between the vector of actual ratings and the vector of predicted values, normalized such that and have the same sum. Then the quadratic weighted κ is calculated as:

The metric typically ranges between 0 (random assignment) and 1 (perfect agreement) but may be negative for models which perform worse than expected by chance. The advantage of this metric over accuracy is that it measures the extent of mismatch. Accuracy treats any misclassification equally while the κ metric penalizes larger deviations from true response levels.

For model evaluation, the whole dataset consisting of 252740 rows was randomly split into 80% train and 20% test sets. In general, models were evaluated with the κ metric using stratified 5-fold cross-validation (CV) on the train set (to ensure proportionate representation of salary levels), followed by training on the whole train set and validation on the test set. In the case of optimized rounding, the mean of optimal rounding coefficients from cross-validation was used in the fit on the whole train set prior to model validation. For classification models, the response variable was converted to string type prior to fitting and mapped onto integer values for evaluation. For reference, further evaluation metrics are also provided for the best model. These include accuracy, confusion matrix and AUC values.

## Light gradient boosting machine (LGBM)

LGBM is a boosting algorithm – one that uses an ensemble of weak learners to generate a strong learner. LGBM is more efficient than related algorithms such as e.g. XGBoost partly due to histogram binning of continuous predictors, and is well-suited for larger data sizes. Unlike many other tree-based modeling approaches, it grows trees leaf-wise rather than level by level and thanks to this it tends to achieve lower loss than level-wise algorithms.

In this project, traditional gradient boosting decision trees were used as base learners in LGBM. The most important hyperparameters to optimize include those controlling tree complexity such as e.g. maximum tree depth. The contribution of each tree to the outcome is determined by the boosting learning rate. Overfitting can be reduced by optimizing the bagging fraction for each tree as well as by tuning L1/L2 regularization.

Initially, default LGBM models were fit to data. The summary of model evaluation can be found in the table below:

|  |  |  |
| --- | --- | --- |
| **Default model type** | **Mean κ score for stratified 5-fold CV on training set** | **Test set κ score** |
| LGBM regressor  (nearest integer rounding) | 0.623 | 0.622 |
| LGBM regressor  (optimized class threshold) | 0.688 | 0.687 |
| LGBM classifier | 0.637 | 0.637 |

The better regressor (one with optimized rounding of salary categories) and the classifier were subjected to hyperparameter tuning. For each of the LGBM models, initial tuning of parameters included a search for the maximum tree depth, learning rate as well as L1 and L2 regularization in a 2000-trial Optuna study. The optimal parameters were as follows:

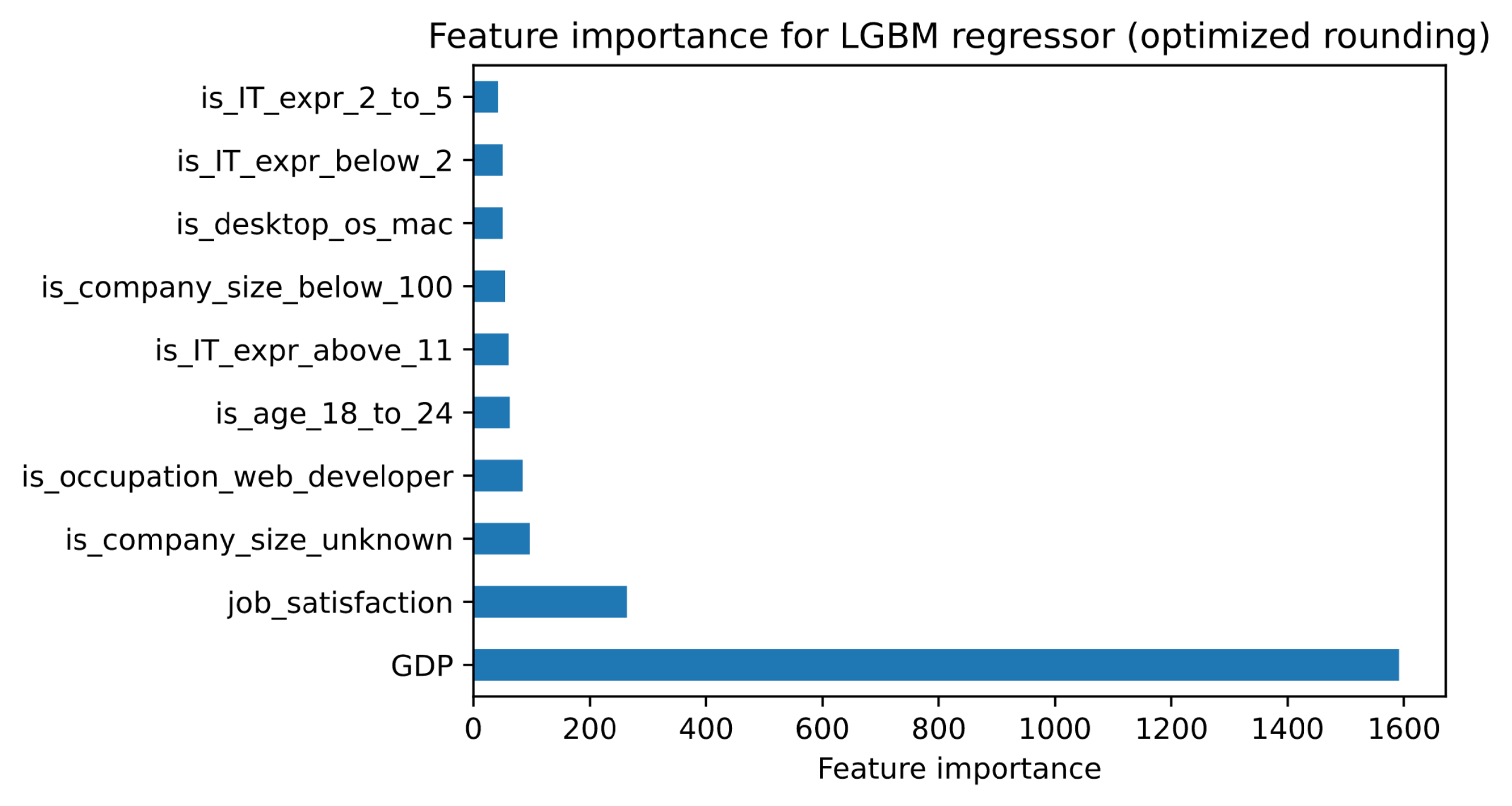
* LGBM regressor with optimized rounding: {'learning\_rate': 0.12752857768803944, 'max\_depth': 9, 'lambda\_l1': 15, 'lambda\_l2': 100}.
* LGBM classifier: {'learning\_rate': 0.16726096140196198, 'max\_depth': 12, 'lambda\_l1': 65, 'lambda\_l2': 40}.

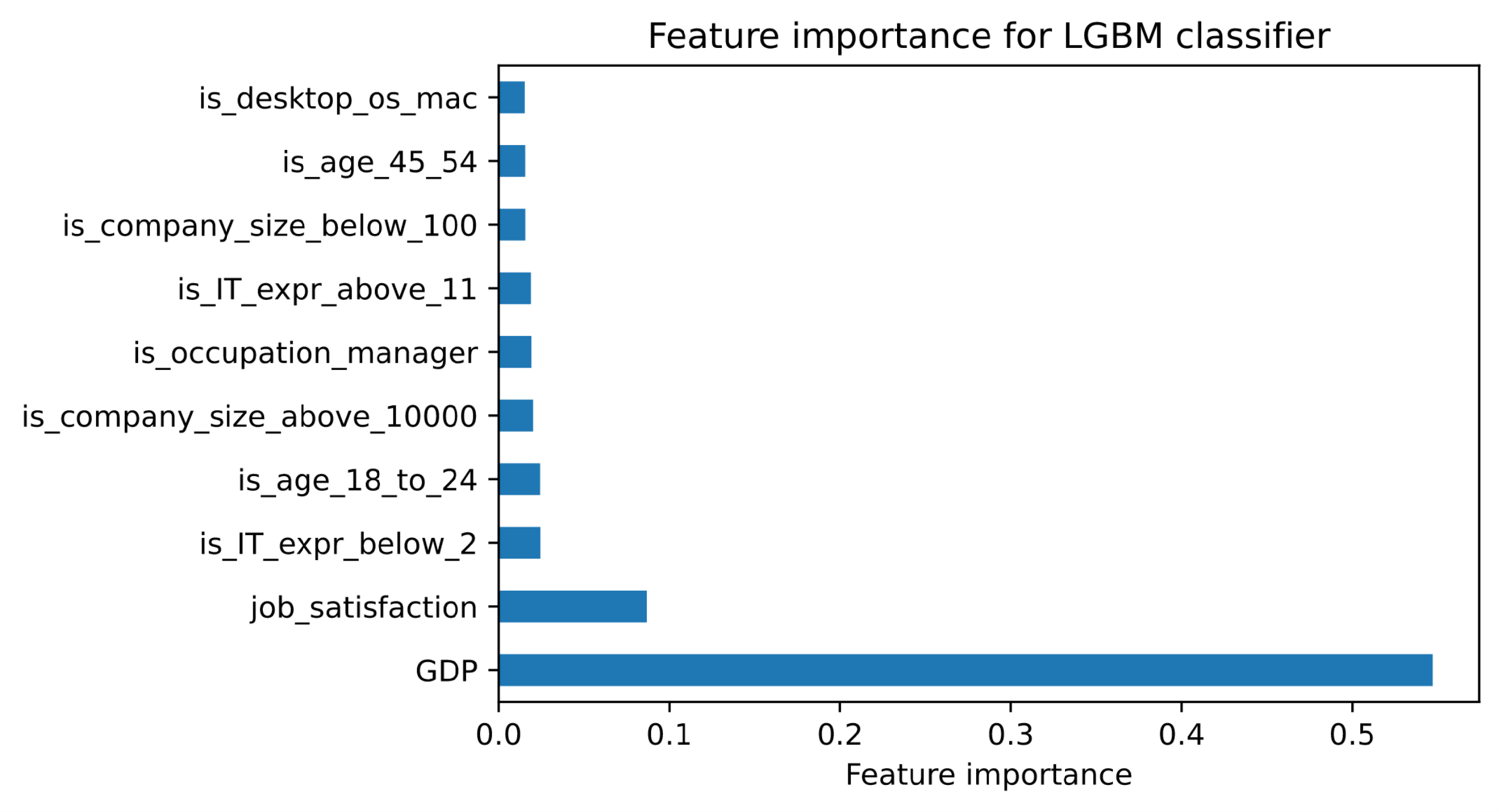
The evaluation scores for the tuned models were:

|  |  |  |
| --- | --- | --- |
| **Tuned model type** | **Mean κ score for stratified 5-fold CV on training set** | **Test set κ score** |
| LGBM regressor  (optimized class threshold) | 0.690 | 0.692 |
| LGBM classifier | 0.610 | 0.618 |

The best performing LGBM model was the regressor with optimized rounding thresholds. Following parameter tuning, its fit improved slightly in both CV and test set validation. The LGBM classifier had a substantially κ lower evaluation metric for the default model compared to the regressors and parameter optimization did not improve the fit of the classifier.

The models are useful as they allow the interpretation of variable importance measures across models, as shown below:





We can observe a strong influence of GDP on developer salary, followed by factors related to e.g. job satisfaction, age and IT experience.​

## Random forests (RFs)

RFs is an ensemble learning algorithm in which many (typically 500-1000) decision trees are built. To decorrelate trees, bagging (bootstrap aggregation) is implemented – each tree is built on a subset of data generated using random sampling with replacement. Another method to minimize tree correlation employed by RFs is the randomization of the subset of features at node splits. The prediction of the model is the mean of the trees for regression and the mode for classification. Like LGBM, RFs can be useful in the interpretation of the influence of variables as the model can produce variable importance measures.

Initially, default RF models were fit to data. The summary of model evaluation can be found in the table below:

|  |  |  |
| --- | --- | --- |
| **Default model type** | **Mean κ score for stratified 5-fold CV on training set** | **Test set κ score** |
| RF regressor  (nearest integer rounding) | 0.677 | 0.680 |
| RF regressor  (optimized class threshold) | 0.676 | 0.686 |
| RF classifier | 0.524 | 0.530 |

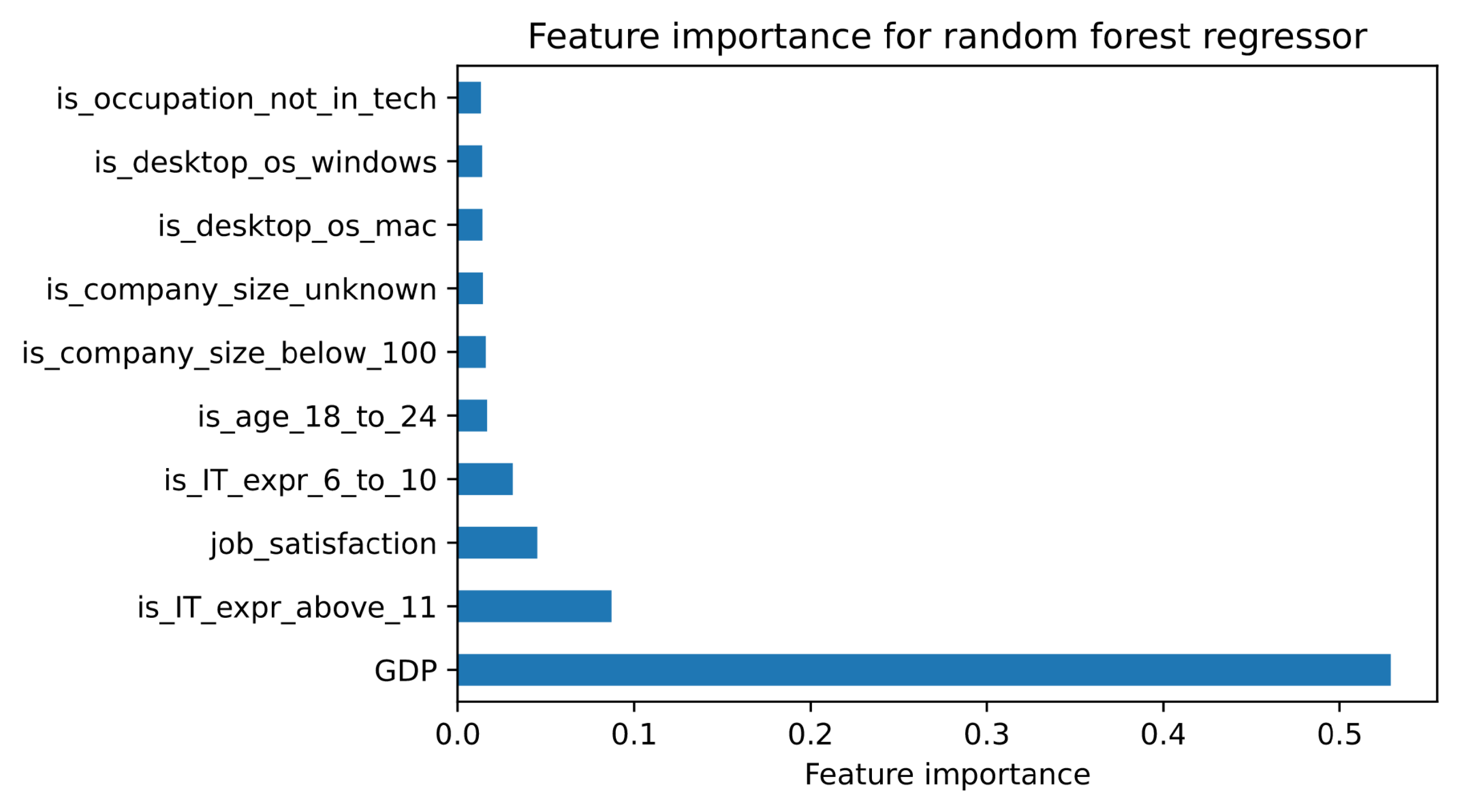
The better regressor (one with optimized rounding of salary categories) and the classifier were subjected to hyperparameter tuning. Some of the main parameters to tune in RFs include the total number of trees, bagging fraction and the number of randomly selected features at each branching step. Ranges of values investigated were between 1 and 30 for tree depth, 25 values between 0.001 and 1 for bagging fraction as well as ‘auto’ (no feature subsampling) and ‘sqrt’ (square root of the number of features) for feature selection at node splitting steps. A grid search for optimal parameters was performed (using the RandomizedSearchCV function, a total of 300 fits) for RF regressor and classifier on a subset of 20000 data points to decrease computation time. The best parameters were used to fit the models on the train set and generate evaluation metrics as before:

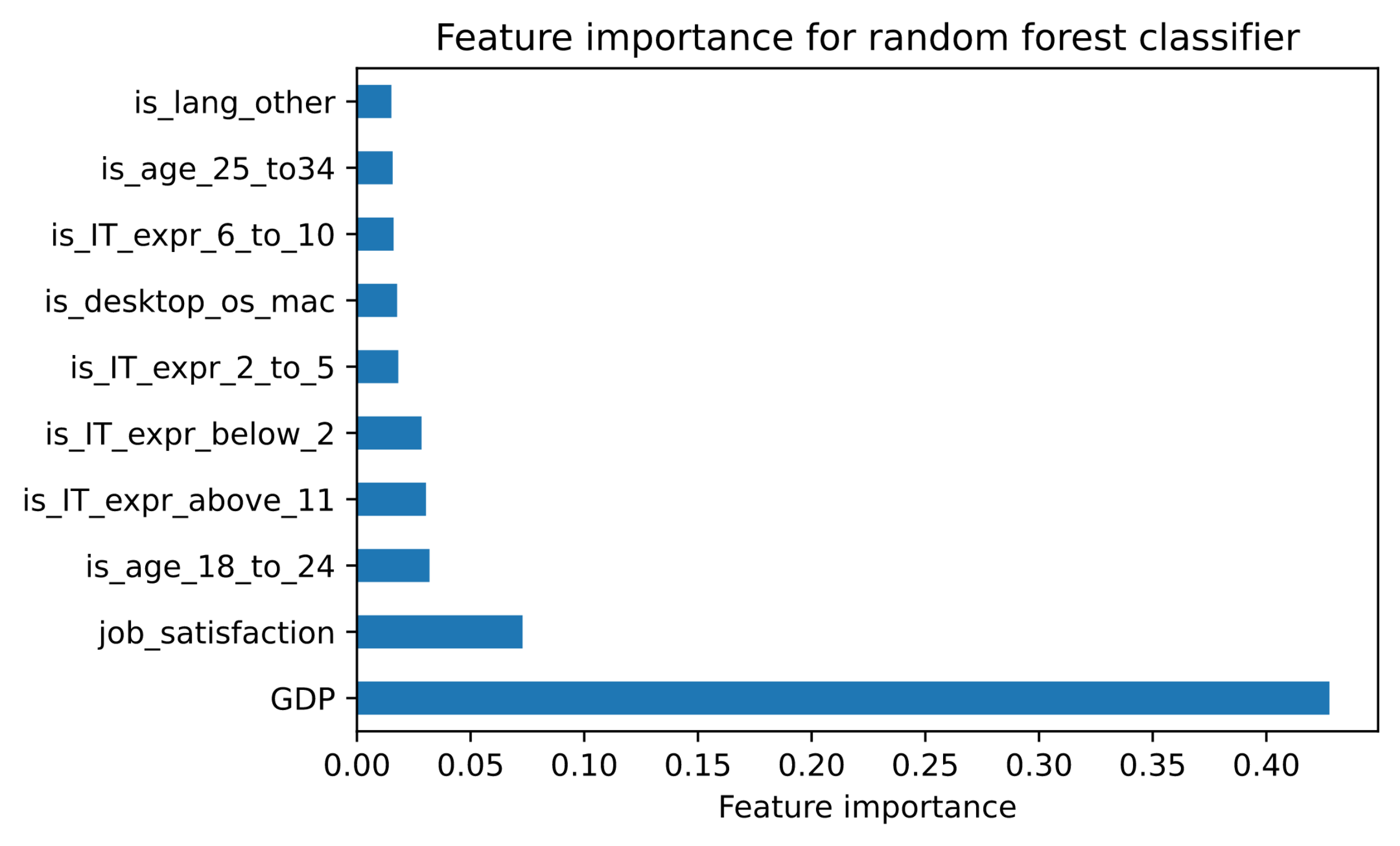
* RF regressor: {'max\_samples': 0.45887500000000003, 'max\_features': 'auto', 'max\_depth': 18}.
* RF classifier: {'max\_samples': 0.7086250000000001, 'max\_features': 'auto', 'max\_depth': 17}.

The evaluation scores for the tuned RF models were:

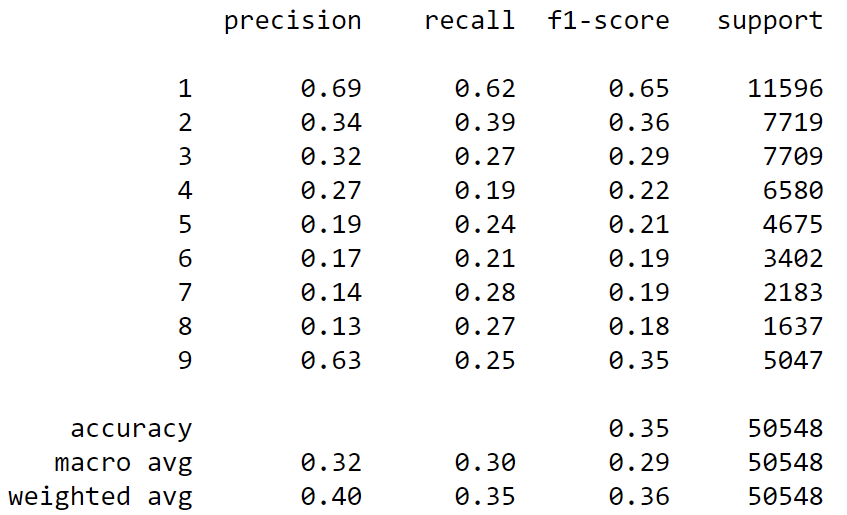
|  |  |  |
| --- | --- | --- |
| **Tuned model type** | **Mean κ score for stratified 5-fold CV on training set** | **Test set κ score** |
| RF regressor  (optimized class threshold) | 0.713 | 0.717 |
| RF classifier | 0.564 | 0.563 |

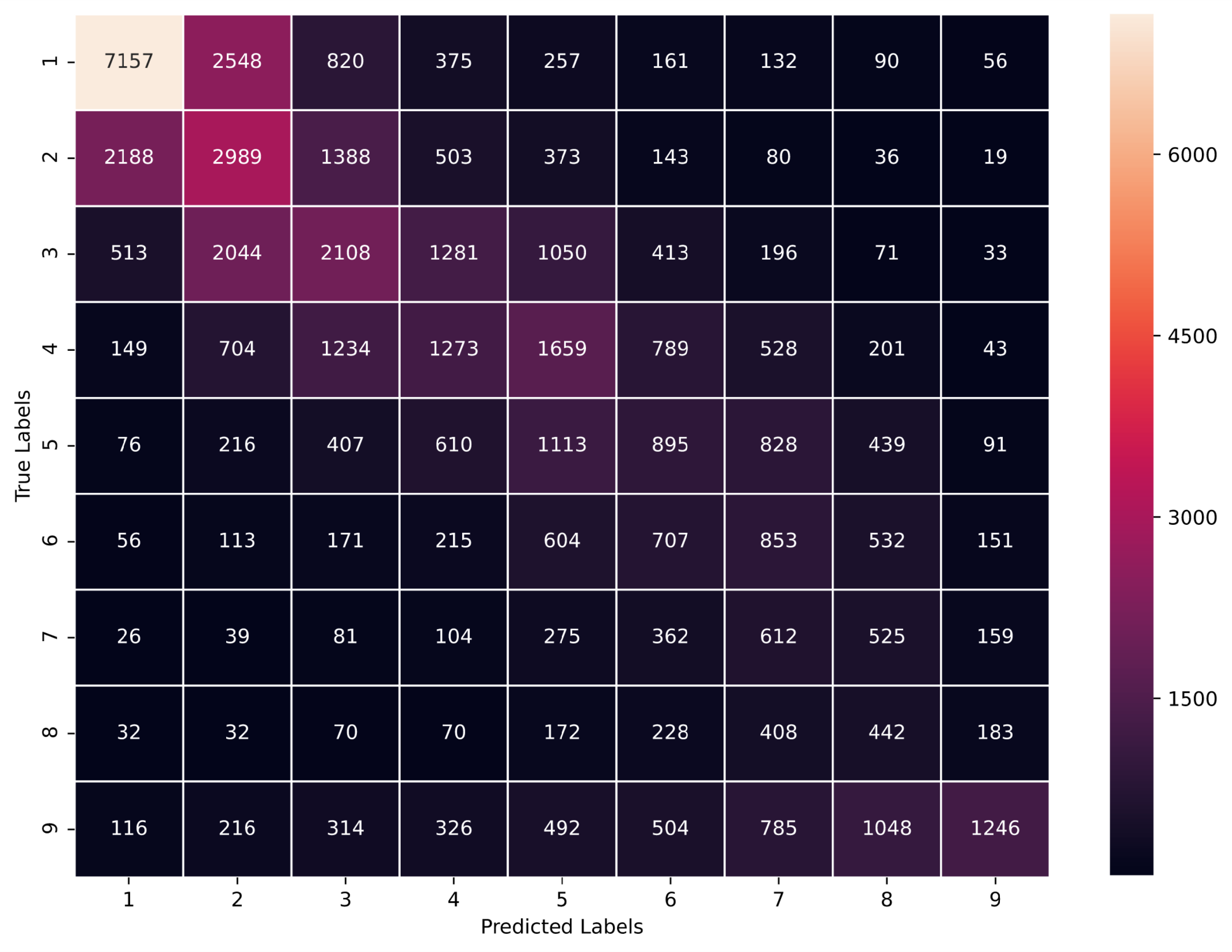
The highest κ score was achieved by the RF regressor with optimized rounding thresholds. Hyperparameter tuning improved CV and test set κ scores of both RF models, though the improvement of the regressor was substantially higher. Variable importance for the two tuned models is reported below:





Given that the parameter-optimized RF regressor with optimized class thresholds was overall the best performing model in the project, several evaluation metrics beyond the κ score were investigated for it. Accuracy of the model was 34.9%. Further metrics are summarized below:



The confusion matrix for salary levels using the best RF regressor was also generated:

The following AUC values were obtained for each salary level:

* 1: 0.768.
* 2: 0.625.
* 3: 0.584.
* 4: 0.557.
* 5: 0.566.
* 6: 0.567.
* 7: 0.601.
* 8: 0.605.
* 9: 0.615.

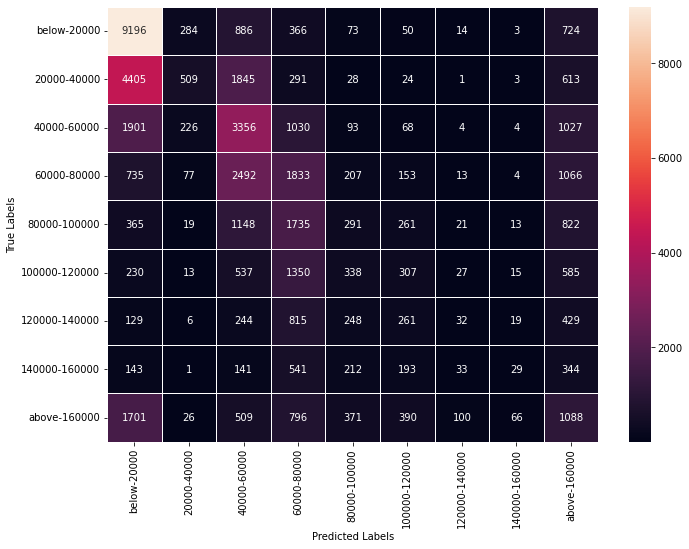
## Logistic Baseline Results

As a baseline for the other models, we created a simple multinomial logistic regression model. We used the logistic regression algorithm available in scikit-learn. Unfortunately, it does not consider ordered response variables, only distinct classes. Salary is thus treated not as a continuous variable, which it is natural to assume it is during the data generating process, but as an unordered class of intervals of $20,000.

Assessing model performance is not a trivial matter. The widely used and familiar metrics fail to account for predictions that are one class away, or on the entirely different side of the salary scale.

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Accuracy | 0.329 |
| Macro averaged precision | 0.250 |
| Macro averaged recall | 0.219 |
| Macro averaged F1 | 0.196 |
| Kappa | 0.382 |

At this point, results are not great, and it looks like we are unable to make reasonable predictions. We investigate the confusion matrix.



The confusion matrix shows us that predictions for low-salary individuals is ok, but we struggle with predictions for high-salary individuals. We notice that the largest classes also have the most false-positives, suggesting class imbalance affects the results. From salaries from $80 000 and up we are barely doing better than by assigning salaries by random.

There are a few problems with using multinomial logistic regression here. One is that the salary is very dependent on which country you are in. Without extensive feature engineering to capture interaction effects and other non-linear patterns the model will not take this appropriately into account. For example, the model lacks the context to find the correct contribution of a knowledge of a particular programming language.

Next is as mentioned that the response variable in our dataset, salary as an unordered class, fail to capture the underlying data generating process well.

The confusion matrix illustrates these problems working in tandem. Almost all places outside of the US any salary above $80 000 is considered a very high salary requiring both expert knowledge of a wide variety of tools and techniques as well as additional skills or responsibilities. While in parts of the US, like Silicon Valley, salaries between $80 000 and $160 000 are within the range of normal salaries. Very high salaries outside of the normal range will largely be due to many idiosyncratic factors. This makes them hard, if not impossible, to predict in a meaningful way. When we mix salaries that are contextually within a normal range with salaries that are contextually beyond it in the same class, an algorithm that finds a linear combination of predictors will struggle. Especially without a significant amount of purposeful additional context from interaction terms and polynomial features.

A multinomial logistic regression model also does not capture how bundles of skills contribute to a salary. A valuable front-end developer might know Java, CSS and HTML, while a data scientist might know R, Python, and SQL. Both occupations might have a similar salary, but a simple regression model will only add the contributions of each skill individually. There are also no diminishing returns to knowing many languages included in the model.

We try other types of modelling to mitigate these issues.

## Further Modelling Techniques

Alternative modelling methods could involve strategies developed specifically for ordinal classification. One approach could involve transforming the ordinal classification task into a series of ordered binary classifiers (Frank and Hall 2001). Another strategy could involve ordinal RFs as developed by Hornung (Hornung 2020) where the RFs use the out of bag (OOB) error estimates during the construction of the forest to tune the interval boundaries of the underlying continuous variable for the ordinal classes. As a complement of the modelling approaches used in this project, neural networks could be used for this salary prediction task, and model stacking, or blending could also be applied.

## Challenges

Given that the data source utilized is from surveys administered annually within a span of 10 years, many data transformation and normalization was required to account for the data variations across the years.

For example, some questions were asked in a select number of years and not others. While we were interested to model the trend for remote work overtime, this question was surveyed in the most recent years, which made it impossible to include it as a key variable in our analysis, and hence could not make predictions for how patterns in remote work have changed pre- and post-COVID-19 pandemic.

Additionally, some of the multi-select questions such as those that asks respondents about their occupation, or the programming languages used, allowed for multiple responses, in some cases as many as 25 different options. This highly inflated the number of unique combinations of responses and added a layer of complexity to the analysis. As such we resorted to only consider the first option that the user selected for the analysis purpose.

Furthermore, there was inconsistent data type for responses of the same questions that were asked across different years. For example, the data type for the question that asks respondents about the number of years of coding experience was categorical in earlier years, and a mix of numerical and categorical values for the latest years.

Non-response bias is another challenge that was unique to the project due to the data type being used i.e. survey data. To tackle this, we applied different techniques to impute the missing values, and in other cases where there was significant data missing, we dropped the null values, an effect which potentially may introduce bias to our results.

In the model that we used, we treated the response variable, the salary earned by a respondent which has an underlying continuous data generating process as a multinomial classification problem due to data quality issues. And did not account for inflation as we had consumed the compensation in the model without including the respective year corresponding to each of the responses. Moreover, in the analysis of our results the GDP was flagged as an important variable, which is attributed to the very different salary levels in different countries. However, its importance is irrelevant for answering our hypothesis.

## Future Iterations

With the time constraint, there are some iterations of the solutions which the team would like to explore in future iterations. Firstly, a time series analysis of how compensation for developer salary has changed over time, as well as a prediction model to predict compensation in future years. This would require a continuous variable for compensation rather than categorical as we have done for the classification problem. Additionally, we did not get to the work from home analysis as we would have liked to in the beginning. This would be analysed using the most recent survey data, as this survey question was only asked in the last few years of the annual survey. We would like to further tune the models that we had created, as well as perform additional data wrangling. For instance, we only selected the first available proficient programming language as well as the first available occupation as these survey questions are multiselect. In future iterations we should be more thoughtful around this and perhaps include the secondary and tertiary languages and occupations.

* + Account for class imbalance
  + Normalize salaries by country / region / GDP
  + More feature engineering

## Summary

### Key Outcomes:

Below are the key outcomes from our modelling and discovery:

* Data quality issues made it reasonable to investigate the problem with a multinomial classification model
* This type of modelling requires nuanced, sophisticated, and less intuitive interpretation of the results
* A kappa score of 0.68 suggest a moderate level of agreement. I.e., a decent, but far from great model \*
* The current dataset and model can be useful as a starting point, but does not today give sufficient results to be used for the projects outlined use-cases
* The only data we collected from another source, GDP, had by far the largest variable importance. Suggesting adding and adjusting for more contextual data for each respondent will be sensible addition

## Supporting Literature

Two interesting and supporting literature pieces to make note of are the following studies: Martin et al. 2018 and Barrerro et al. (2021), both of which are aligned very closely with our problem statement and project context.

### Martin et al. 2018

In a study by Martin et al. (2018) the researchers analysed 4,000 job offers from Spanish IT recruitment portal to determine the most rewarded components of job descriptions for candidates and recruiters. The group used ensemble models to classify salary ranges and found that the best classifiers were voting and random forest models which achieved an accuracy of approximately 84%. Furthermore, they found that workers could be clustered based on skillsets and that experience was more rewarded than education. Clustering via skillset versus education level may be a consideration of methodology we would consider in a future iteration.

### Barrero et al. (2021)

An analysis of working from home adjustments of over 30,000 Americans from Barrero et al. (2021) suggests that the COVID-19 pandemic will have a lasting impact on working arrangements. An estimated 20% of full workdays are expected to be supplied from home after the pandemic ends. Companies spending 5-10% less money in major city centres and an implied relative productivity boost of 5% in the post-pandemic economy could be expected to drive changes in the salary of developers due to re-optimized working arrangements, but these effects remain unexplored. In future iterations of this problem, work from home attributes in later years as that is a survey question found within the dataset towards the end of the decade.

# Works Cited

* Barrero et al, 2021 <https://www.nber.org/system/files/working_papers/w28731/w28731.pdf>
* <https://www.ijert.org/predictive-analysis-of-hr-salary-using-machine-learning-techniques>
* <http://www.actuaries.org/pbss/colloquia/helsinki/papers/koskinen.pdf>
* Martin et al, 2018 <https://www.atlantis-press.com/article/25899235.pdf>
* <https://insights.stackoverflow.com/survey/2021#salary-and-experience-by-developer-type>
* <https://www.pewresearch.org/fact-tank/2022/06/15/in-the-u-s-and-around-the-world-inflation-is-high-and-getting-higher/>
* <https://deliverypdf.ssrn.com/delivery.php?ID=772029127089022106125085005096110029127088006054089053094031103111099117030093065027041060048030012023098000114072081064002077033006063021080090085075031119121087109058013055087111104121067000105094070099099081017005088091003028017078100027096114020003&EXT=pdf&INDEX=TRUE>
* <https://www.pwc.com/gx/en/research-insights/economy/global-economy-watch/projections.html>
* Frank, Eibe, and Mark Hall. 2001. “A Simple Approach to Ordinal Classification.” In , 145–56. https://doi.org/10.1007/3-540-44795-4\_13.
* Hornung, Roman. 2020. “Ordinal Forests.” *Journal of Classification* 37 (1): 4–17. https://doi.org/10.1007/s00357-018-9302-x.