Machine Learning for Smarter Hiring Algorithms

Ben Gigone

I. Project Definition

1. Project Overview

How do we select potential employee candidates to ensure their future success and maximize longevity within our company? While a solution to this problem may seem impossible given the inability to truly simulate how an individual will behave in the future, advancements in machine learning and data analysis can help us approach a reasonable result. As a recruiter within a large North American sales company, an aspect of my job is to identify and determine the potential success and longevity of an individual based on a relatively limited amount of information. Essentially, I am tasked with attempting to determine what an individual may achieve in a set number of years (in terms of both individual sales and potential income), while also assessing the likely "lifespan" of their career. Through machine learning, past employee data can help with the prediction of the success of future candidates. In addition to predicting "success", this information will also aid in highlighting features (i.e. location, previous employment, etc.) that may have contributed towards an individual's success. This allows for the identification of specific markets and their corresponding results, which will lead to more efficient hiring techniques.

2. Problem Statement

The purpose of this project is to create and implement a machine learning algorithm to classify the potential success of new representatives based on a specified number of features. Using a dataset of approximately 2500 representatives annually earning $100,000 or more, this algorithm will seek to match characteristics of potential new employees with current high-earning representatives to determine these future income levels. By analyzing market locations, previous employment, and spousal status, this algorithm will be able to estimate the success of an applicant. Additionally, this algorithm will provide insight into the successes and failures of current market locations, and will aide in determining markets that should be focused on.

3. Metrics

The metric used to evaluate the precision of the algorithm's prediction is the F1 score method. While a simple accuracy score may be more intuitive, this project calls for an evaluation method capable of accurately evaluating data with imbalances in the dataset. As the dataset classes are heavily imbalanced, the selected evaluation method should be relatively insensitive to these imbalanced data classes.

The F1 score computes a result based on the precision (number of true positive results divided by the number of all positive results), p, and recall (number of true positive results divided by the number of positive results that should have been returned), r. Essentially, the F1 score is an average of both the precision and recall, with an high value of 1 (perfect precision and recall) and a low value of 0 [1]. Physically, the F1 score is calculated as,

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

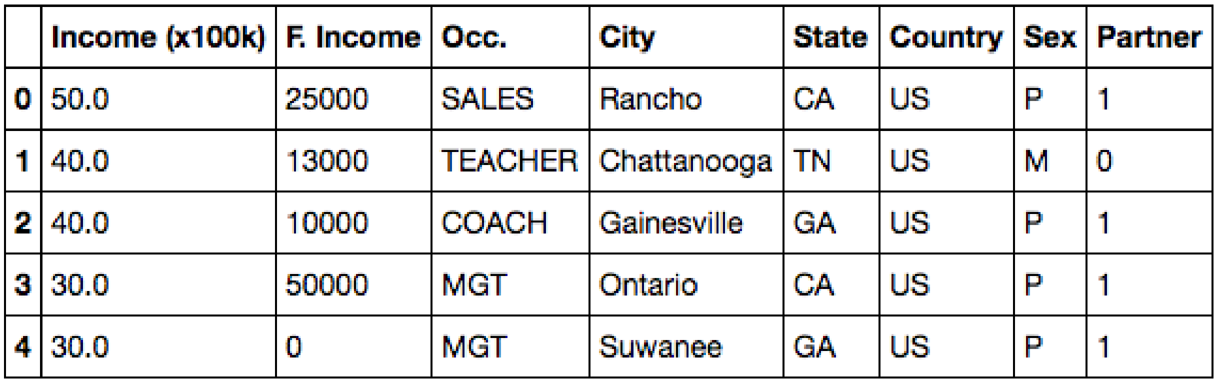
II. Analysis

4. Data Exploration

In this project, a dataset containing the information of 2578 representatives is examined. Specifically, the information of each individual within the dataset is made up of 8 main features:

- Income Level

This feature is the current income level of the representative. Income levels within this dataset (and corresponding frequency of each) are $100,000 - $249,999 [x1951], $250,000 - $499,999 [x405], $500,000 - $749,999 [x115], $750,000 - $999,999 [x31], $1,000,000 - $1,999,999 [x59], $2,000,000 - $2,999,999 [x11], $3,000,000 - $3,999,999 [x2], $4,000,000 - $4,999,999 [x2], $5,000,000 + [x1]. Within the dataset, the income level is represented by multiplying the value income level by 100,000. For example, the $100,000 - $249,999 range is represented as 1, the $2,000,000 - $2,999,999 range is represented as 20, and so on. This feature is extremely important, as the categorization of new information into this feature is essentially the desired output. The data used in this work takes the following shape (first five rows of the dataset):



	Income (x100k)	F. Income	Occ.	City	State	Country	Sex	Partner
0	50.0	25000	SALES	Rancho	CA	US	P	1
1	40.0	13000	TEACHER	Chattanooga	TN	US	M	0
2	40.0	10000	COACH	Gainesville	GA	US	P	1
3	30.0	50000	MGT	Ontario	CA	US	P	1
4	30.0	0	MGT	Suwanee	GA	US	P	1

- Former Income

This feature is the former income of the representative in question. Within the dataset, the former income is directly the value shown. This information is important as it will help to categorize the previous income levels of potential new representatives into sections based on the historical data.

- Former Occupation

This feature is the former occupation of the representative in question. This feature is important as it will allow for the determination of fields that produce individuals capable of success in this new position. For example, we will be able to visualize how those transitioning from a career in teaching compare with those transitions directly from academia. Within the dataset, specific occupations have been split into abbreviated subsets. A legend specifying the abbreviations of each occupation is available within the dataset sheet.

- City, State, Country

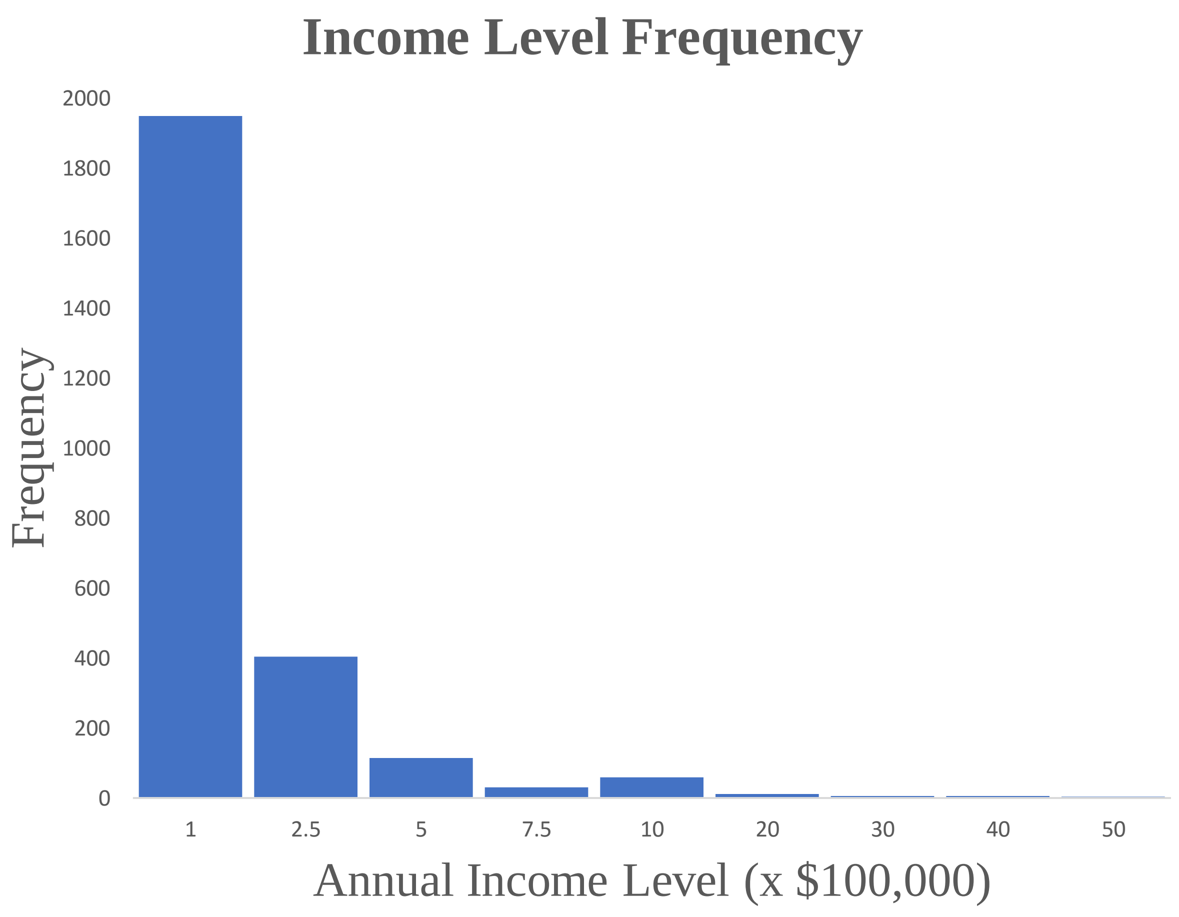
These three features represent the current city, state, and country in which the representative resides and works. This information is listed as is in the dataset. These features are extremely important as they will aide in the determination of successful markets, and where future focus should be applied.

- Sex, Spousal Status

These two features represent the sex and marital status of the representative. These features will aide in the categorization of potential new representatives, and will allow for correlations to be drawn between the effect of partnerships on potential income levels. In the dataset, the three given inputs for sex are M, F, and P (male, female and partnership). Additionally, to individually determine this affect, a final partnership dataset column contains a 1 if a partnership is present and a 0 if the representative is listed as an individual.

5. Exploratory Visualization

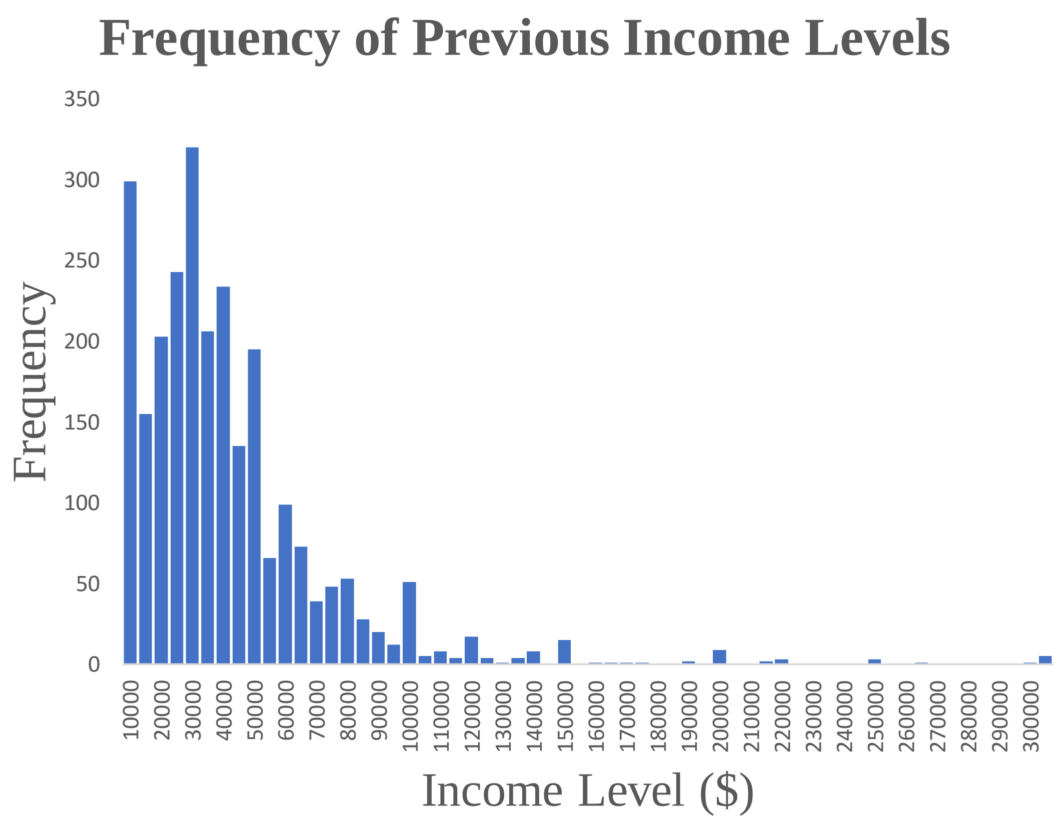
Figure 1 below depicts the frequency at which each income level is represented in the data. Visualizing this relationship confirms the previous assumption describing how the dataset is imbalanced. This imbalance between classes confirms the selection of metrics previously described. We can clearly see from this plot that the lower annual income levels (particularly $100,000 and $250,000) are represented much more heavily than higher income levels (i.e. $3,000,000 - $5,000,000).



From the above figure, it can be noted that:

- The majority of current employees (1951) are currently making $100,000 a year.
- The higher income levels (i.e. $2 million, $3 million, $4 million, and $5 million) are home to drastically fewer individuals (11, 5, 5, and 4, respectively).

Figure 2 below depicts the frequency of income levels of individuals before working at the financial company. As evident, these data points are more evenly spread out, and categorize typical average incomes quite well. It is also clear, however, that a weighted analysis technique must be used to scale this information during the categorization process.



6. Algorithms and Techniques

This project takes advantage of a number of machine learning methods to successfully classify “new hires” into a potential future income level class. Regarding specific algorithms and techniques, two classification based supervised-learning algorithms are used.

Firstly, the classifier used is a support vector machine classifier, typically used in similar category assignment scenarios. This type of supervised classifier works by defining a separating hyperplane to categorize new examples, given some labeled training data. Support vector machines were chosen for this project due to its high accuracy and efficiency on smaller, cleaner datasets. As this dataset is relatively small and has clearly defined classes (i.e. no noise due to overlap), support vector machines are a clear choice. To optimize the classifier, the following parameters may be tuned:

- “C”: The penalty parameter, C, should be tuned in the event that the input data is unbalanced, as in the present case. Tuning this parameter ensures that misclassification does not occur. A low C value makes the decision surface smooth, while a high C value attempts to classify all training examples correctly [2].
- “gamma”: The parameter gamma defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected [2].

Secondly, a Random Forest classifier will be used to complement the SVM classifier previously discussed. A Random Forest classifier is an ensemble learning method which operates by creating a number of decision trees and outputting the class that is the mode of the specified classes. This method is able to provide optimal results for even the most complex classification tasks. The Random Forest model will nicely complement the results gathered from the applied support vector machine. To optimize the Random Forest Model, the following main parameters may be tuned [3]:

- "n_estimators": specify the number of decision trees in the forest.
- "max_features": determine the number of features to consider when determining the best split.
- "max_depth": determines the maximum depth of the tree.

Regarding preprocessing, simple data rescaling (via an algorithm such as MinMaxScaler) will be conducted on the Income feature to ensure the data is normalized. Additionally, binarization of features such as Marital Statues (i.e. 0 – single, 1- partnership) will also be conducted. If necessary, preprocessing will be conducted to eliminate outliers that are clearly skewing the final results.

7. Benchmark

To create an initial benchmark for the classifier, a basic random forest classifier was used on the dataset. The random forest classifier was used "out-of-the-box", and no parameter tuning was used to optimize the classifier. After splitting the test and training data (using 15% of the data to test on), initial predictions using this basic classifier resulted in an f1-score of 0.873.

III. Methodology

8. Data Preprocessing

The data preprocessing done before training and testing our data consists of a few key steps:

1. The target variable “income” is separated from the other features in our dataset.
2. A “MinMaxScaler” is used on the “income” variable. As this variable is highly skewed, this preprocessing step allows for the normalization of this numerical input. This ensures that these highly-skewed inputs don’t negatively affect our classification methods.
3. The data points are all randomized to prevent inaccuracies and reduce data bias.
4. As each data point is clearly defined, all inputs are now one-hot-encoded. This process assigns a numerical value to each categorical input.

Once these preprocessing steps have been completed, the data is now ready to be split into training and testing sets.

9. Implementation

During this stage, the inputted data was preprocessed using the process outlined previously. Once preprocessed, the data was split into training and testing sets. For this project, the training set was defined to by 85% of the dataset, while the remaining 15% was set aside for testing. Following the splitting of the data, a

Random Forest Classifier was fit to the data as our benchmark model. As mentioned previously, this benchmark produced an F1-score of 0.873. It is important to note that the F1-score will be used for determining the success of classifiers within this project. While a simple accuracy score may be more intuitive, this project calls for an evaluation method capable of accurately evaluating data with imbalances in the dataset. As the dataset classes are heavily imbalanced, the selected evaluation method should be relatively insensitive to these imbalances. The algorithms used in the Jupyter Notebook accompanying this document are well commented and clearly laid out.

Next, an SVC was fit to the data to be compared to the benchmark. The SVC produced an optimized F1-score of 0.87. A number of parameters were tuned when fitting, and the default SVC setting provided this optimized result. As this performance is worse than the benchmark F1-score produced by the “out-of-the-box” Random Forest Classifier, the Random Forest Classifier will be used as the final classifier.

Regarding specific implementation, the fitting of classifiers and validation testing follow the same method:

```
from sklearn.ensemble import RandomForestClassifier


clf = RandomForestClassifier()
clf.fit(X_train, y_train)
```

First, the classifier (in this case, Random Forest) is imported. Once defined, the classifier is fit to the training data (X_train and y_train).

```
from sklearn.metrics import f1_score
```

```
pred = clf.predict(X_test)
score = f1_score(y_test, pred, average = None)

print ("Random Forest F1-score is {}".format(round(score[0], 3)))
```

Next, the method for evaluation (in this project, the F1-score parameter) is imported. We then use our classifier’s predict function on the X_test variable (set to be 15% of the data in this case). Finally, the F1-score is calculated based on the weighted average of both the precision and recall, taking y_test, and our previous prediction of X_test (i.e. “pred”) as input. Training the optimized Random Forest, the SVC, and performing validation all follow the same general method.

Regarding challenges experienced, upon a first attempt, only the categorical values were one-hot encoded. This lead to problems further along, as fitting the classifiers to this dataset provided many errors. Finally, all inputs (including the target variable) were one-hot encoded. This resulted in a multi-class target variable, so the selection of a particular classifier (i.e. Random Forest Classifier) was necessary.

10. Refinement

During the classification optimization, all available parameters of the Random Forest Classifier [4] were tuned to fit a classifier with the highest possible F1-score. Important parameters in refining this model were;

- n_estimators: this parameter determined the number of trees in the forest. Essentially, increasing this parameter’s value increases the number of distinct decisions the classifier sets to make before converging on a solution.

- max_depth: this parameter sets the “depth” of the tree. Specifically, the nodes are continually expanded until each “leaf” is pure, or until all leaves are found to be less than a specified number of split samples.
- min_samples_split: this parameter specifies the number of samples required to split an internal node, and consequently affects the max_depth parameter.
- min_samples_leaf: this parameter sets the number of samples required to be at a leaf node. Typically, this parameter is set as a percentage of the overall training set.
- bootstrap: This parameter determines whether or not bootstrap samples are used when building trees. This statistical method aims to increase the overall accuracy of the classifier.
- random_state: This parameter says that the seed used in produced by a random number generator. This ensures that biases in the representation of the data don’t negatively affect the outcome of the classifier.

By tuning this classifier, a final F1-score of approximately 0.891 was achieved. While a number of other parameters were available to test, they were often dependent on parameters used here, and resulted in lower F1-scores than the benchmark model provided. It is clear that optimization of parameter has produced a more robust model capable of more accurately classifying a new input.

IV. Methodology

11. Model Evaluation and Validation

As mentioned in the previous refinement section, a number of parameters were tuned to achieve an optimal result. The following parameter values were assigned to our Random Forest Classifier and were subsequently applied to the dataset:

- n_estimators: Produced highest results at approximately 30 estimators.
- max_depth: Optimized at a maximum depth of 30.
- min_samples_split: Showed no effect on F1-score. Remained at default value of 2.
- min_samples_leaf: Produced an adverse effect on F1-score as it was increased. Remained at default value of 1.
- max_leaf_nodes: Showed no effect on F1-score. Remained at default value of 0.
- bootstrap: By not using bootstrap samples when building trees (i.e. bootstrap = false), a higher F1-score was achieved.
- random_state: Showed no effect on F1-score. Remained at default value of 0.

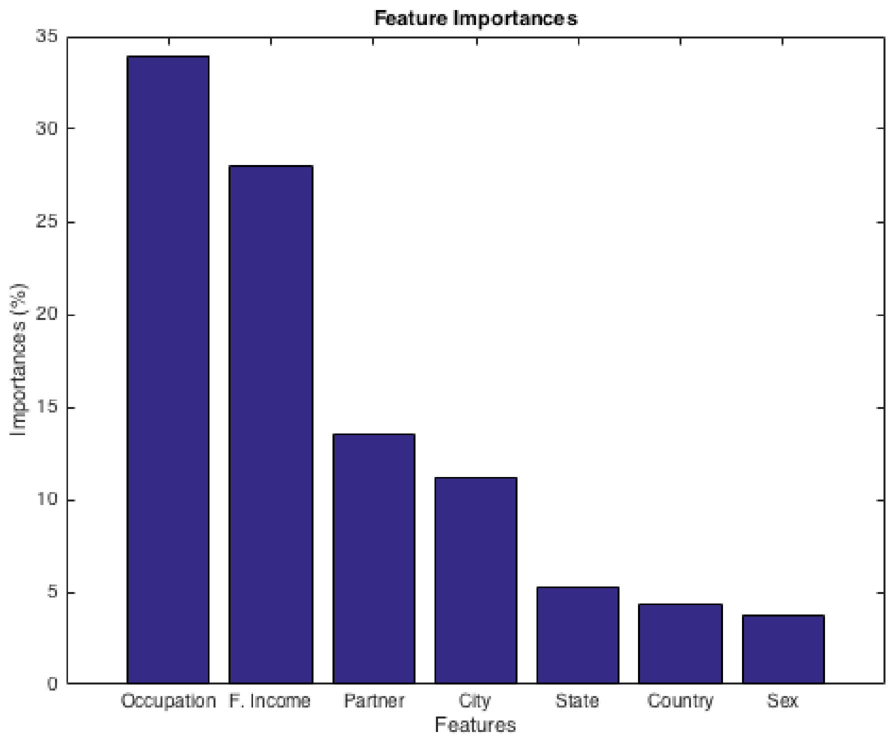
To verify the robustness of the model, it is important to test the algorithm on a new, independent dataset. In this project, a K-Fold cross-validation technique was used to split the original dataset into some new dataset with training and testing portions. As a completely new dataset is not available, this method will separate the data so that it is independent from the previously trained dataset. For this method to be validated, we can compare the corresponding F1-Score from our optimized random forest model with our new dataset split by the K-Fold validation technique.

Method	F1-Score
KFold	0.88
Random Forest	0.891

We can see that the F1-Score gathered in this validation case is close to the F1-score resulting from the original Random Forest model. This further validates that our result is able to accurately classify new information.

12. Justification

To further justify the results of income level predictions given a new potential candidate, analyzing the feature importance stored within the Random Forest Classifier allows us to determine which features most frequently correspond to a higher income level. Since the features were one-hot encoded, the separated features were manually collected to determine the output from the classifier;

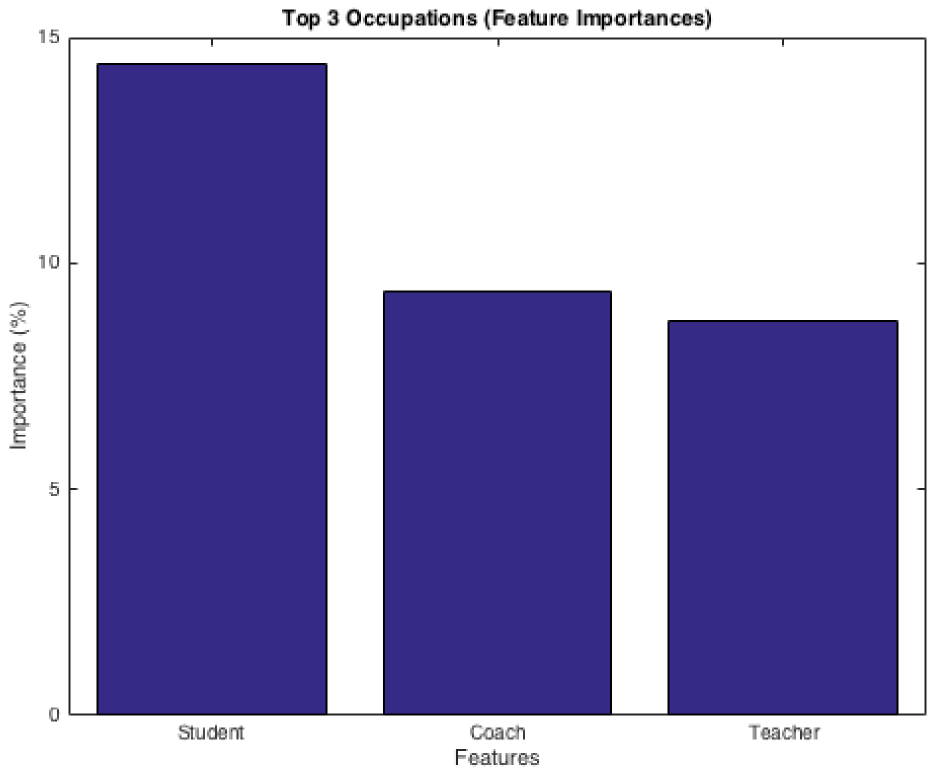


From this figure, we see that occupation and former income (corresponding to approximately 34% and 28%, respectively) are the most relevant indicators of the potential income level of a new candidate. Interestingly, the next most important indicator (roughly 14%), is the marital status of the individual. It is also clear that among the features used to attempt to classify a new individual, the residence location and sex of the individual are relatively unimportant.

V. Conclusion

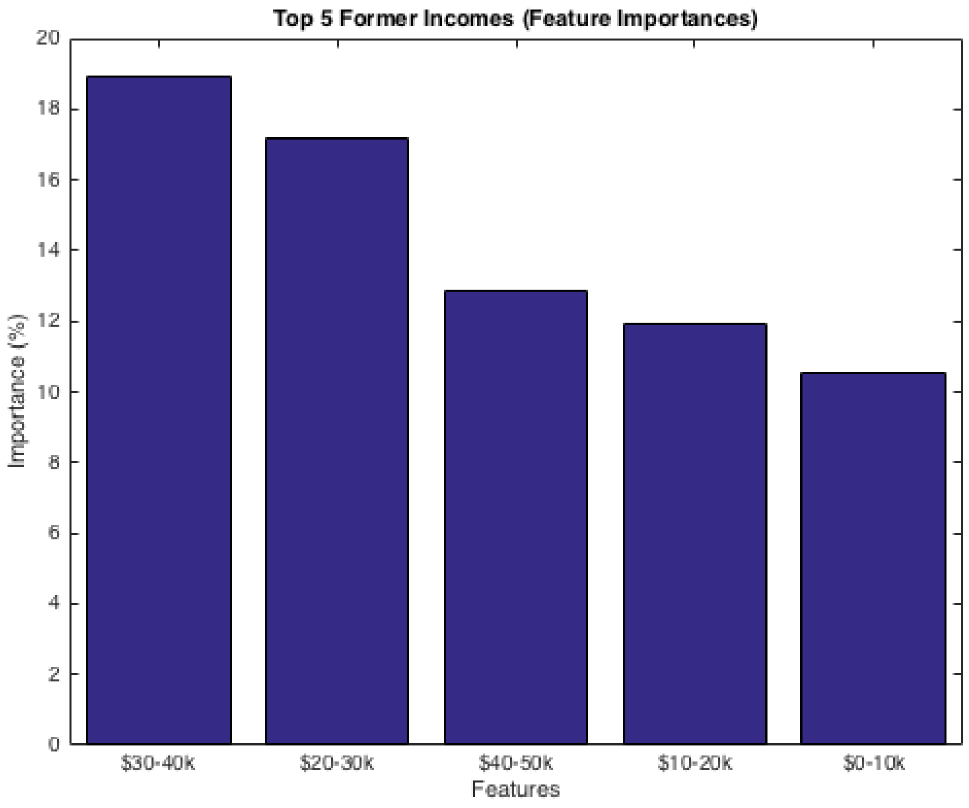
13. Free-Form Visualization

By further analyzing the feature importance of a number of feature categories, we are able to draw further conclusions from our data. As the goal of this project was to determine the markets that should be focused on when hiring new candidates, our results tell us a number of things. Firstly, from the broad feature importance figure shown in Section 12, we see that focusing on location of residence and/or sex of the individual is not as relevant as the individual's former occupation, income, and marital status.

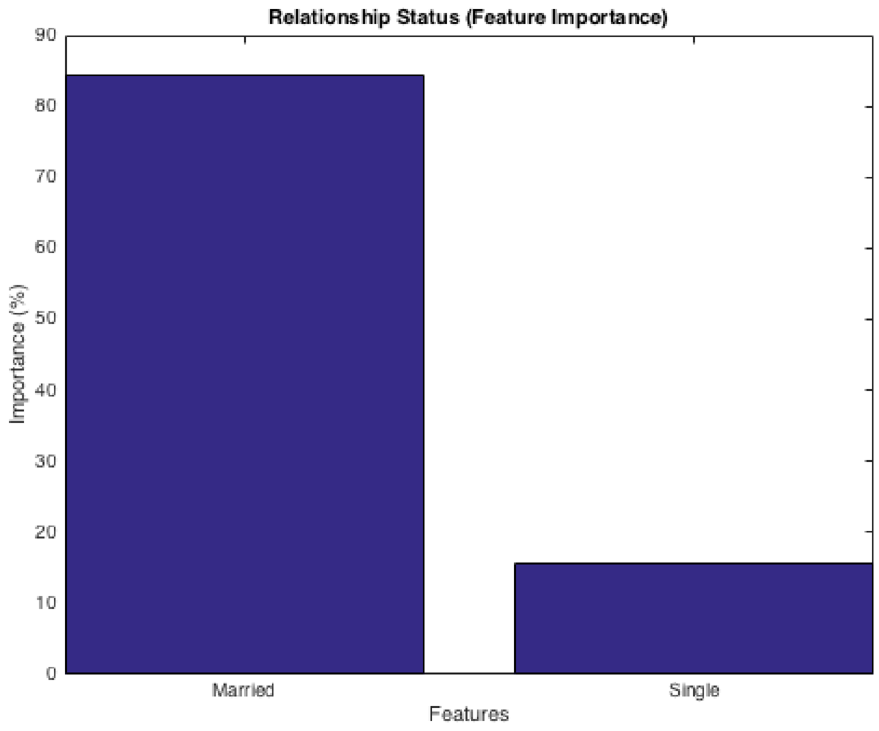


From this figure, we see that the top 3 former occupations indicative of a higher future income level are; 1. Student, 2. Coach, and 3. Teacher. Together, these three occupation categories (out of 33 inputted categories) represent a feature importance of approximately 31%. As such, we can determine that someone from the field of education will most likely succeed in our company**.

** Interestingly enough, when I presented this information to a number of our field trainers they were not surprised in the least. As this career requires certain leadership and teaching abilities, this information lined up with what our current representatives were seeing in the field.



When analyzing the top 5 former incomes, it is initially clear that while a number of high-earning representatives (i.e. 80k+) were considered in the sample data, individuals with former incomes of less than $50,000 annually proved to become more successful at our company. This data also validates the top 3 former occupation figure, as these income levels generally coincide with teachers/coaches ($30 – 50k) and students ($0 – 20k). I believe this information to be the most important gained from this project. This data shows that a high-earning individual will not always be the most successful in our company. While we often attempt to hire individuals with high levels of experience, it is often those with lower incomes (or no incomes, in the case of most students) who succeed the greatest at our company.



Finally, we can see that 86% of the people earning more than $100,000 in our company are married. While the implications of this may run on a more psychological level, it seems that having a partner in business with you (whether physically, or in a more supportive role) clearly leads to higher income.

14. Reflection

The overall process used throughout this process can be summarized as follows:

1. A problem was discovered, and a relevant dataset was gathered.
2. Basic preprocessing was done to prepare for machine learning techniques.
3. A benchmark was created to test the classifier.

4. The classifier was trained with the data and further turned until a set of parameters helped to reach an optimized result.
5. The important features were extracted and analyzed to form a solution to the initial problem.

Steps 4 and 5 were the most difficult for me, as I had to dive much deeper into the aspects of one-hot encoding a number of features, as well as how to analyze the feature importance values output from these one-hot encoded values. While steps 4 and 5 proved to be most difficult, they also have become the most interesting aspects of the project for me. This further knowledge on how to interpret one-hot encoded feature importances proved invaluable to me throughout this process. As such, this project has provided extremely valuable information for me and my company, which will readily be used in our hiring processes.

15. Improvement

As far as potential improvement, the only obvious note regards the user-friendliness of the result. Without an in-depth knowledge of interpretation of one-hot encoded features, a result may not initially be apparent. As such, further development of a program capable of reverting one-hot encoded features back into their initial features (after running through the classifier) would allow for much simpler interpretation.

Bibliography:

[1] “Sklearn.metrics.f1_score.” *Sklearn.metrics.f1_score — scikit-Learn 0.19.0 documentation,* Scikit-Learn, scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

[2] “1.4. Support Vector Machines.” *1.4. Support Vector Machines — Scikit-Learn 0.19.1 Documentation*, scikit-learn.org/stable/modules/svm.html.
learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.
[3] “Sklearn.ensemble.RandomForestClassifier.” *Scikit-Learn 0.19.0 documentation*, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html