Predicting success for a bank telemarketing call

Siddarth Reddy Karuka

Final Project – Python for Data Science

Abstract

Telemarketing calls require a lot of resources, has very little success ratio, and can be annoying. Banks often make such telemarketing calls to advertise and to get customers to subscribe for their products. This project aims to predict the success of these telemarketing calls using the data available from UCI Machine Learning Repository. The project uses four different classification models, compares their outcomes and provides suggestion on model to be used. It is found that either Logistic Regression or k-neighbors classifier can be used depending on the business needs.

Motivation

- Telemarketers make an average of 300-500 calls in an eight-hour day
- Banking institutions are no exception
- Telemarketing calls are annoying when we receive too many of them

What if we can predict the success of a call?

- We can reduce the number of calls by calling only the ones that are likely to subscribe
- This means less calls for the telemarketer, and less annoying calls for the people who might not be interested – a win win situation that saves money and time for everyone involved

Dataset

- This project uses “Bank Marketing Data Set” from the UCI Machine Learning Repository.

Dataset information:

> *“ The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. ”*

- This dataset contains 41188 data points, with 20 input variables, and one output variable

Variables involved

Input variables:

- Bank client data:

  1 - age (numeric)
  2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
  3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
  4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
  5 - default: has credit in default? (categorical: 'no','yes','unknown')
  6 - housing: has housing loan? (categorical: 'no','yes','unknown')
  7 - loan: has personal loan? (categorical: 'no','yes','unknown')

- Related with the last contact of the current campaign:

  8 - contact: contact communication type (categorical: 'cellular','telephone')
  9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
  10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
  11 - duration: last contact duration, in seconds (numeric)

- Other attributes:

  12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
  13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
  14 - previous: number of contacts performed before this campaign and for this client (numeric)
  15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

- Social and economic context attributes

  16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
  17 - cons.price.idx: consumer price index - monthly indicator (numeric)
  18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
  19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
  20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):

21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

Data preparation and exploration

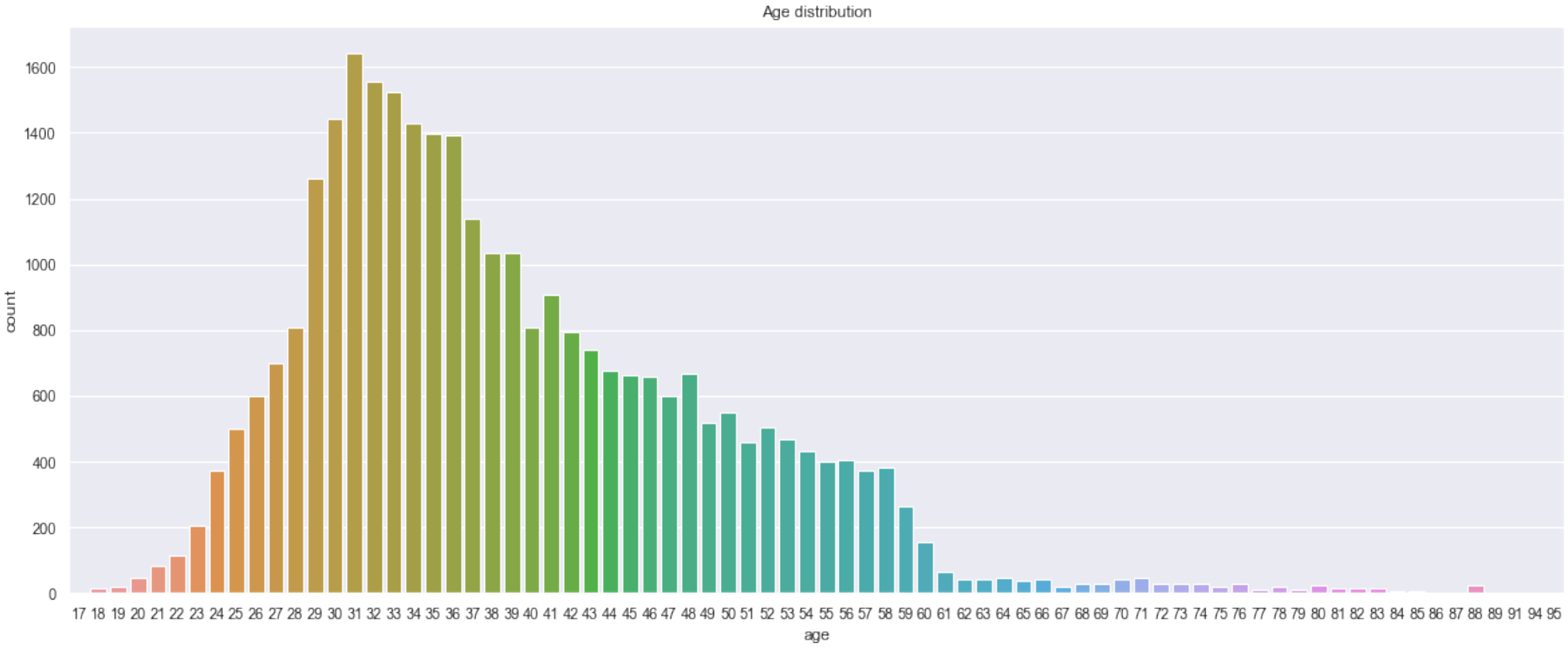
- Doesn't contain any empty fields, but has several "unknown" values in various fields
  - Deleted 10700 out of 41188 rows that contain "unknown" values

- Check for types of variables
  - Contains object, int64 and float64

- Look at the distribution of various parameters (graphs in the next few slides)
  - Age, Job, Marital status, Education
  - Check to see how much of our data has credit in default, has housing or personal loan
  - Duration of the telemarketing call

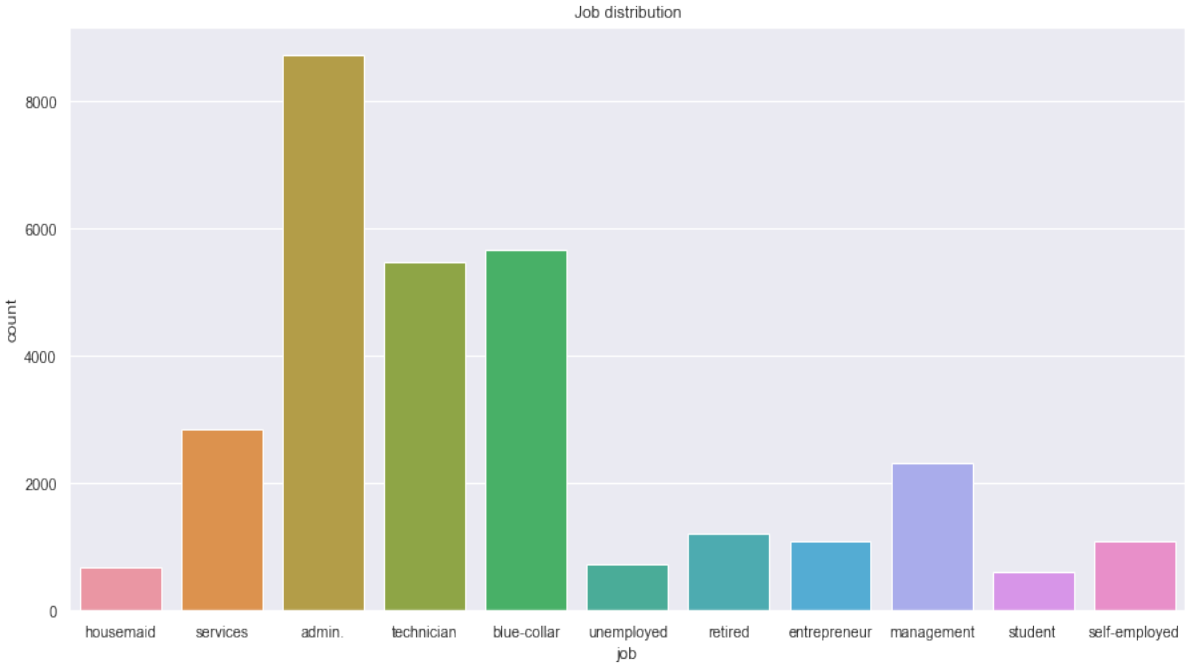
  (This basic exploration helped identify minor issues like having calls with zero seconds duration, which doesn't make sense when telemarketing)

Exploratory figures

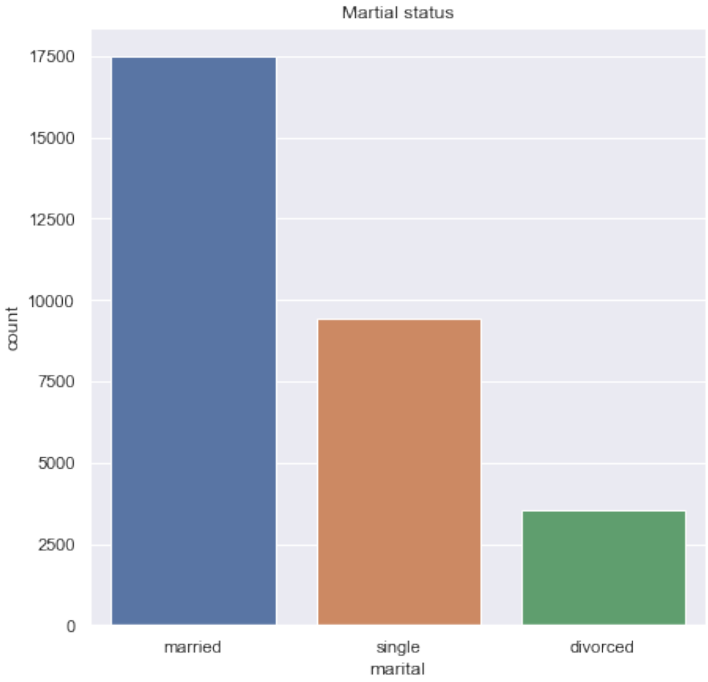


Age distributions shows that there's very few people above age 60 (probably the bank did not contact many people who retired), and none below age 18 (may be age restriction for opening an account). It looks like the bank targeted mid-career people

Exploratory figures

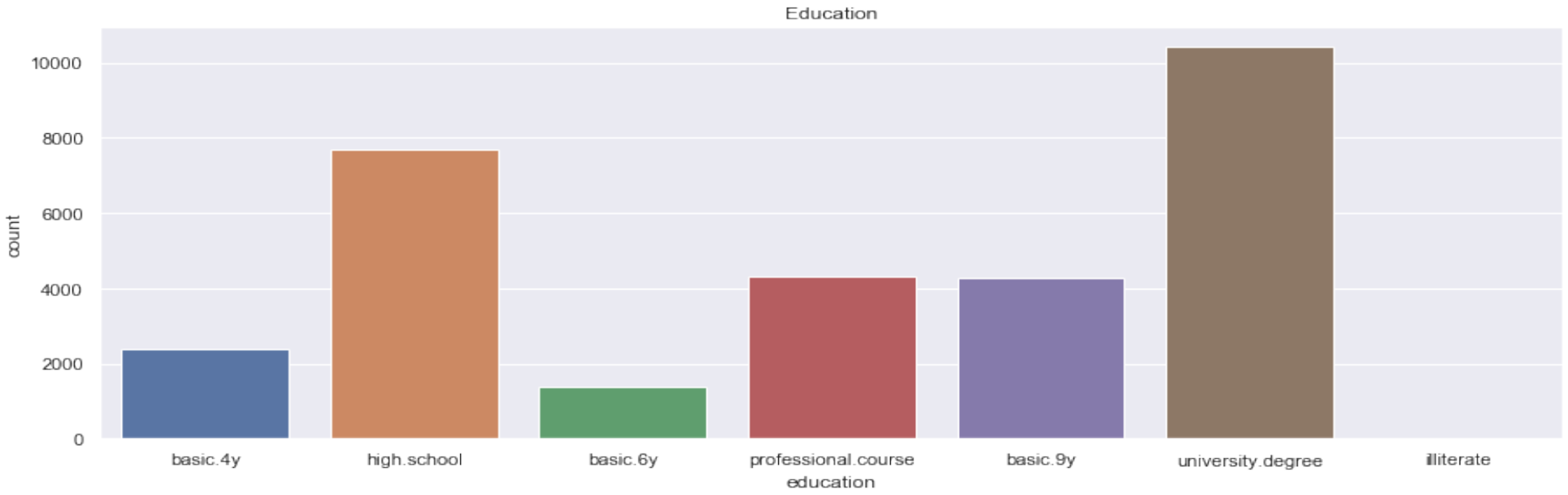


Most people who were contacted have administrative jobs, where as people who were unemployed or students or housemaids were on the lower end of the spectrum



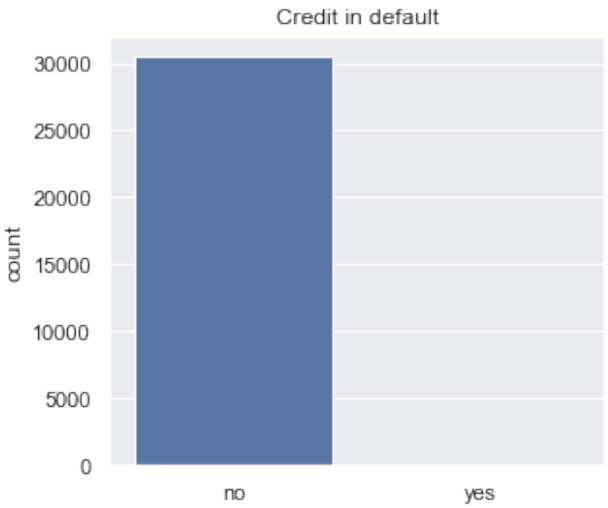
There are more married people in the database than singles and divorced combined.

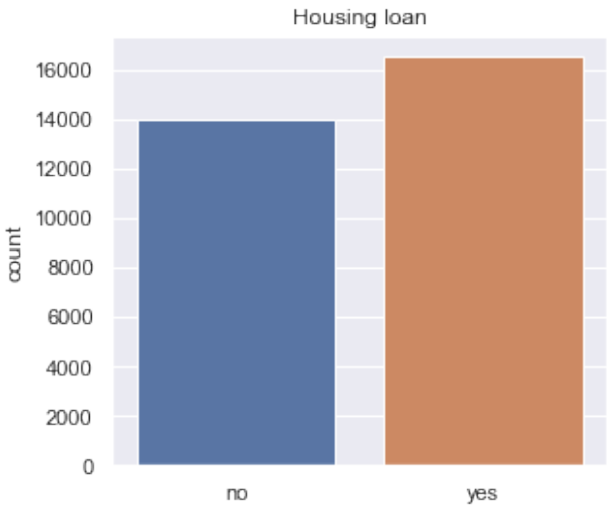
Exploratory figures

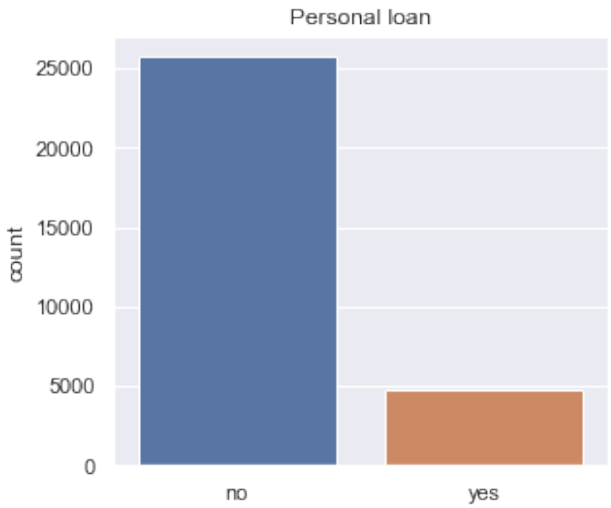


More than 10000 people had a university degree, where as only 11 people were illiterate

Exploratory figures

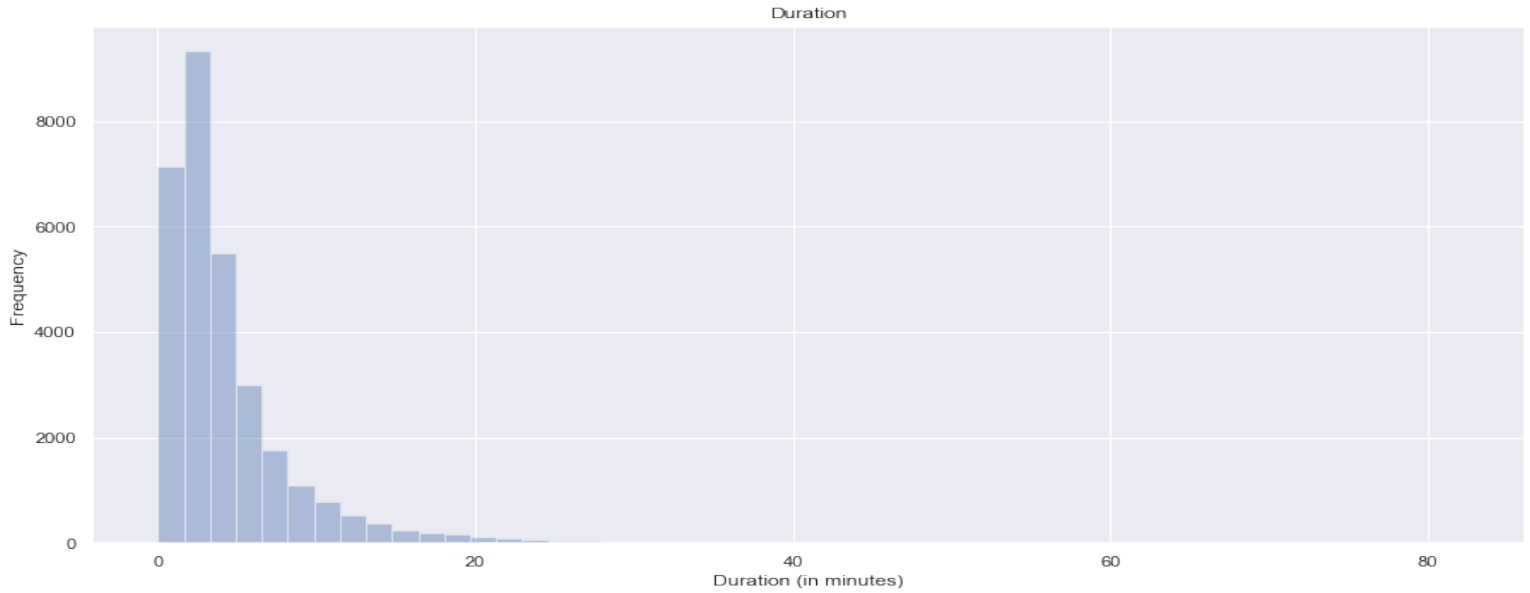






- Only 3 out of 30488 people had their credit in default
- More than half of the people had housing loan
- Very few people have a personal loan

Exploratory figures



The average call duration for these telemarketing calls is 4.32 minutes. The longest one was around 82 minutes, and the shortest reported ones are 0 seconds (4 entries with 0 seconds, not exactly sure how this can be classified as a telemarketing call – these four records are dropped)

Call duration isn't really a predictive variable as we won't know the value of this variable until the call is made, and we will know the output (success or failure) after the call. The "duration" variable is thus dropped out of the dataset

Research Questions

- Can we predict the success of a telemarketing call with the given parameters?
- If so, what model can we use and how much better would it be than the current scenario?

Methods - details

- Predicting success is a binary classification problem.

- Four classification models are employed and the outcomes are discussed
  1. Logistic Regression classifier
  2. k-neighbors classifier
  3. Random Forest classifier
  4. Decision Tree classifier

- Categorical variables were converted to numerical variables (using get_dummies) as most machine learning models can only deal with numerical values.

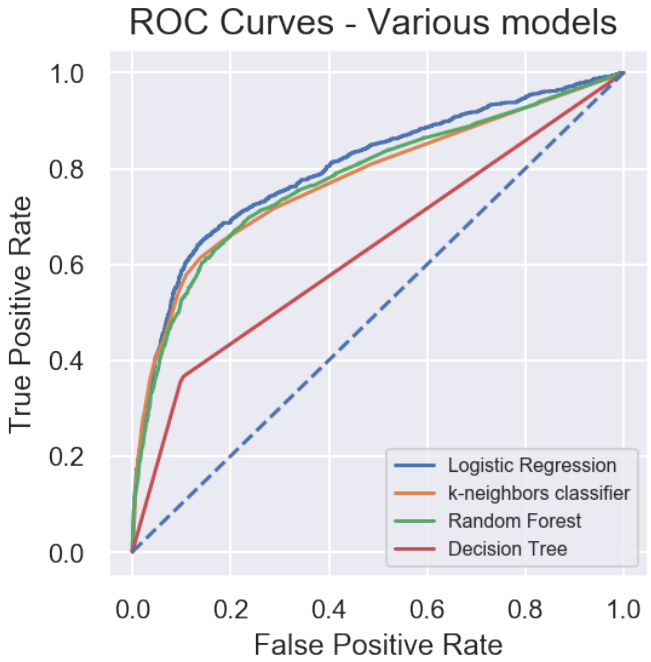
- The dataset is divided into training (75%) and testing (25%) sets, and the classification models are built upon the training set and used on the testing set to check for their performance.

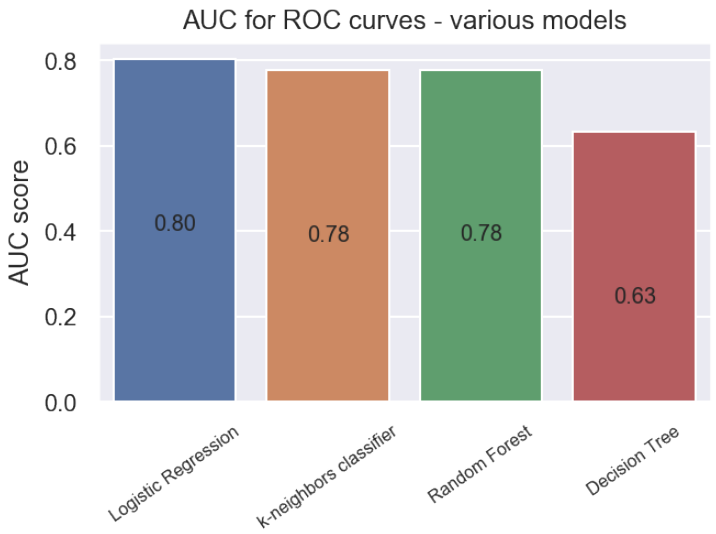
- Metrics like AUC-ROC score, accuracy score, and confusion matrix are used to decide among the models

Methods - appropriateness

- Although there are several more classification models, only a few basic ones are used in this project.
- Machine learning models, such as the ones used here, have optimum performance when the dataset is a balanced one. However, the dataset we have here is imbalanced. (Only 12.6 % success)
- An appropriate fix to this problem would be to collect more data. But that's not a possibility for this project.
- An alternative would be to under-sample or over-sample the training data set. Under-sampling causes loss of information, so over-sampling is used here (Refer to SMOTE for more details)
- Converting categorical variables to numerical can be done in various ways (LabelEncoder or OneHotEncoder or get_dummies). This project uses get_dummies. Further discussion on appropriateness of this can be found in the documented ipython notebook
- Given variables were not on the same scale, so they were scaled using RobustScaler so that the predictive model can attribute equal weights to every variable.

Findings

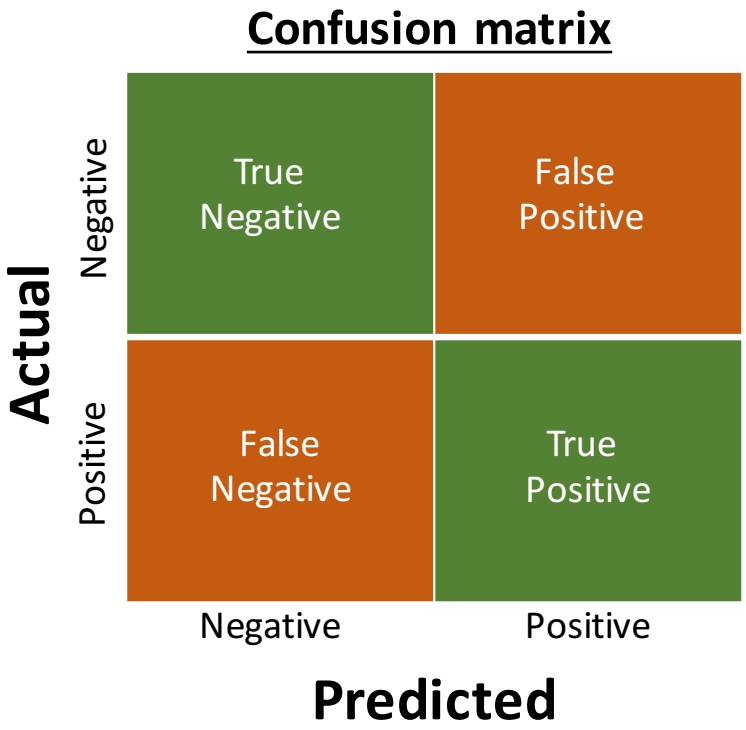




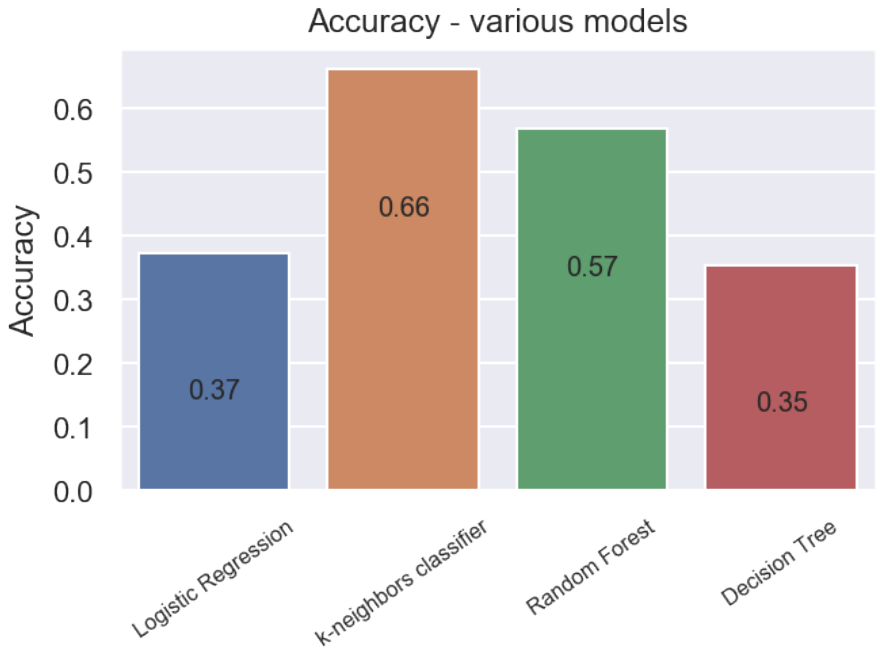
AUC score	Grade
0.90 – 1.00	Excellent
0.80 – 0.89	Good
0.70 – 0.79	Fair
0.50 – 0.69	Poor
0.00 – 0.49	Fail

An ROC (Receiver Operating Characteristic) curve tells us the ability of a model to distinguish true positives from false positives. More specifically, we can look at AUC (Area Under Curve) for this ROC and decide how good the model is. A 100% inaccurate model would have an AUC score of 0, a 100% accurate model would have an AUC score of 1, and a random distribution by chance would yield an AUC score of 0.5

In the above models, Logistic Regression has the best AUC score, but very close to k-neighbors classifier and Random Forest. The AUC score of Decision Tree classifier tells us that it isn't a good model for our project.



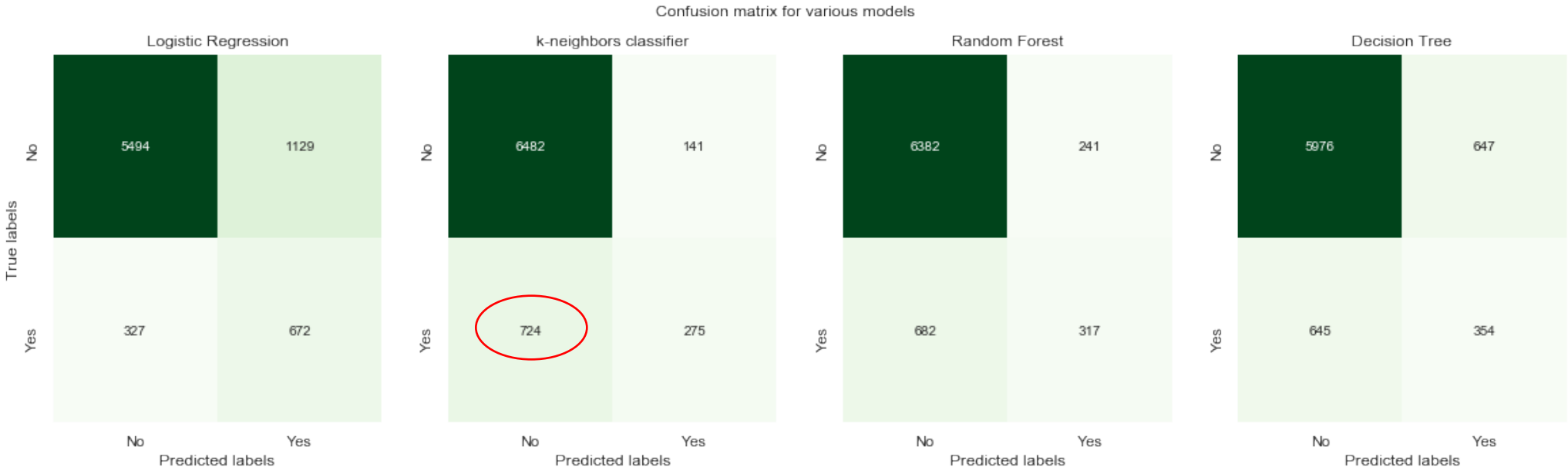
$$Accuracy = \frac{True\ Positive}{True\ Positive + False\ Positive}$$



A confusion matrix is another metric that's regularly used for judging the performance of a model. The above figure indicates that k-neighbors classifier is the most accurate among the four models we have.

Further thoughts on model selection

Selecting a model just based on accuracy can cause harm in certain situations. In this particular situation, where a bank tries telemarketing to get a customer to subscribe for fixed term deposit, it is important to not lose customers.



In the previous slide, it was shown that the k-neighbors classifier is the most accurate among all the models. But if one chooses to go with the predictions of this model, the model would falsely predict the 724 customers (circled above) as people who wouldn't subscribe to the bank product, where as they were actually going to subscribe. So it is also important to take these losses into account when choosing a model

Comparing statistics

	Actual (test dataset)	Logistic Regression	K-neighbors classifier
Total calls made	7622	1801	416
Success %	13.1 %	37.3 % (~3 fold increase)	66.1 % (~5 fold increase)
Total duration of all calls	549 hours	130 hours (76% reduction)	30 hours (94% reduction)
# of customers / accounts gained	999	672 (32% loss)	275 (72% loss)

Limitations

- As shown in the previous slides, there's a trade off between accuracy and loss in customers
- A more sophisticated model (such as Neural Network) could probably perform better, so that's something that can be done as an extension to this project in the future
- These models were built upon an imbalanced dataset, so the appropriateness of these models can be questioned, but can be addressed as more data is collected.
- Various metrics are available for comparing models, so there might be more suitable metrics than the ones chosen here

Conclusions

So what model to use?

- The answer depends on various other factors. Further information is required to make this decision.
  - Is the business trying to expand?
  - What's the long term benefit from gaining a customer?
  - Is it trying to save money?
  - Does it cost more in telemarketing for 100 hours than it costs more in gaining 397 customers?
  - Would the business be more interested in not making unwanted telemarketing calls to save it's image?

- An appropriate response for the first two questions might require a Logistic Regression model, where as a focus on saving money/time/image based on the answers to the last three questions would require a k-neighbors classifier model

References

1. https://www.quora.com/How-many-calls-do-telemarketers-make-in-a-day
2. https://archive.ics.uci.edu/ml/datasets/Bank+Marketing?package=regsel&version=0.2
3. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
4. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
5. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
6. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
7. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html
8. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
9. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
10. https://en.wikipedia.org/wiki/Confusion_matrix
11. https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html
12. https://seaborn.pydata.org/
13. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
14. http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html
15. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
16. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
17. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html
18. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html