Predicting Kaggle Users Tenure

Abstract

To predict an event from historical data is not a new field in science, for the past two decades the data scientist has being trying to create models to make this task simpler. It led to the creation of two new branches in data science i.e., Supervised and Unsupervised learning algorithm.

In this assignment we used some of these techniques to review machine learning concepts such as Exploratory Data Analysis, data munching, data wrangling and finally applying them to the below machine learning algorithm to analyze their performance:

1. Logistic Regression
2. Linear Discriminant Analysis
3. K Neighbors Classifier

Objective

To find a public dataset or machine learning competition and use machine learning techniques to analyze the data.

Introduction

For this assignment, I have selected Kaggle ML and Data Science Survey of 2017 dataset.

The dataset can be found in the below link [Kaggle ML and Data Science Survey, 2017]

Kaggle conducted an industry-wide survey to establish a comprehensive view of the state of data science and machine learning. The survey received over 16,000 responses and they learned a ton about who is working with data, what's happening at the cutting edge of machine learning across industries, and how new data scientists can best break into the field.

Machine Learning study

From the survey data collected by Kaggle, We are going to determine the Tenure/Work experience of a Kaggle user. We are going to use the following columns to determine the experience:

1. GenderSelect
2. Country
3. Age
4. EmploymentStatus
5. CodeWriter
6. CurrentJobTitleSelect
7. MLToolNextYearSelect
8. MLMethodNextYearSelect
9. LanguageRecommendationSelect
10. FormalEducation
11. MajorSelect
12. Tenure
13. FirstTrainingSelect
14. LearningCategorySelftTaught
15. LearningCategoryOnlineCourses
16. LearningCategoryWork
17. LearningCategoryUniversity
18. LearningCategoryKaggle
19. LearningCategoryOth

Methodologies

Logistic Regression

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

In logistic regression, the dependent variable is binary or dichotomous, i.e. it only contains data coded as 1 (TRUE, success, pregnant, etc.) or 0 (FALSE, failure, non-pregnant, etc.).

The goal of logistic regression is to find the best fitting (yet biologically reasonable) model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a *logit transformation* of the probability of presence of the characteristic of interest:

$$logit(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \ldots + b_kX_k$$

where p is the probability of presence of the characteristic of interest. The logit transformation is defined as the logged odds:

$$odds = \frac{p}{1-p} = \frac{probability\ of\ presence\ of\ characteristic}{probability\ of\ absence\ of\ characteristic}$$

and

$$logit(p) = \ln\left(\frac{p}{1-p}\right)$$

Rather than choosing parameters that minimize the sum of squared errors (like in ordinary regression), estimation in logistic regression chooses parameters that maximize the likelihood of observing the sample values.

Linear Discriminant Analysis

Linear discriminant analysis (**LDA**) is a generalization of **Fisher's linear discriminant**, a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.

k-nearest neighbors algorithm

In pattern recognition, the ***k*-nearest neighbors algorithm** (***k*-NN**) is a non-parametric method used for classification and regression.[1] In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- In *k-NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

k-NN is a type of [instance-based learning], or [lazy learning], where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all [machine learning] algorithms.

Code with documentation

Initial Study of data set

```
response.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16716 entries, 0 to 16715
Columns: 228 entries, GenderSelect to JobFactorPublishingOpportunity
dtypes: float64(13), object(215)
memory usage: 29.1+ MB
```

```
print('The total number of respondents:',response.shape[0])
print('Total number of Countries with respondents:',response['Country'].nunique())
print('Country with highest
respondents:',response['Country'].value_counts().index[0],'with',response['Country'].value_counts().values[0],'respondents')
print('Youngest respondent:',response['Age'].min(),' and Oldest respondent:',response['Age'].max())
```

```
The total number of respondents: 16716
Total number of Countries with respondents: 52
Country with highest respondents: United States with 4197 respondents
Youngest respondent: 0.0  and Oldest respondent: 100.0
```

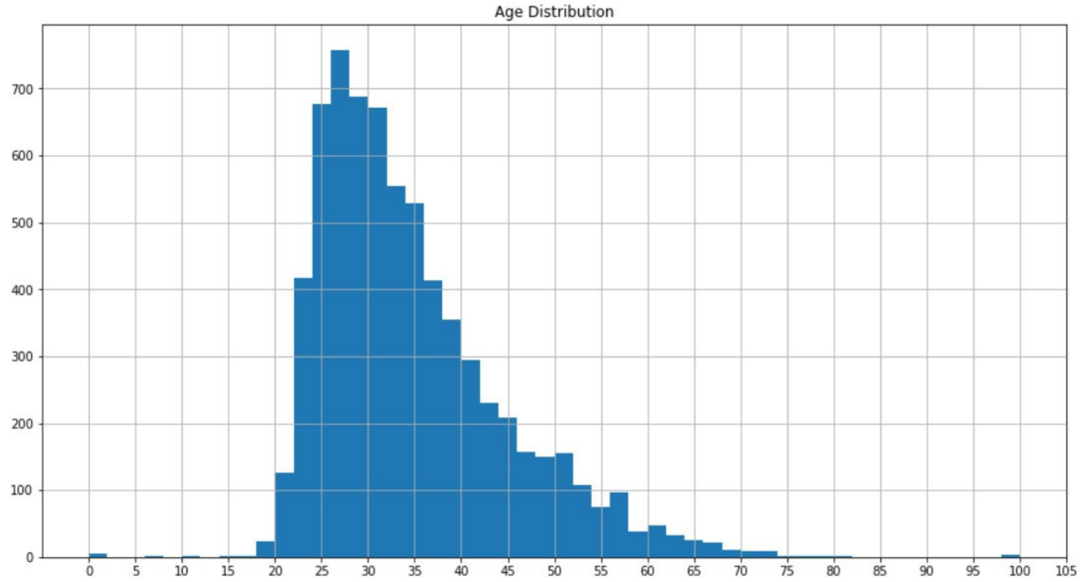
Selecting the appropriate columns

```
response =
response[['GenderSelect','Country','Age','EmploymentStatus','CodeWriter','CurrentJobTitleSelect','MLToolNextYearSelect','MLMethodNextYearSelect','LanguageRecommendationSelect','FormalEducation','MajorSelect','Tenure','FirstTrainingSelect','LearningCategorySelftTaught','LearningCategoryOnlineCourses','LearningCategoryWork','LearningCategoryUniversity','LearningCategoryKaggle','LearningCategoryOther']]
response.info()
reponse_1 = response
```

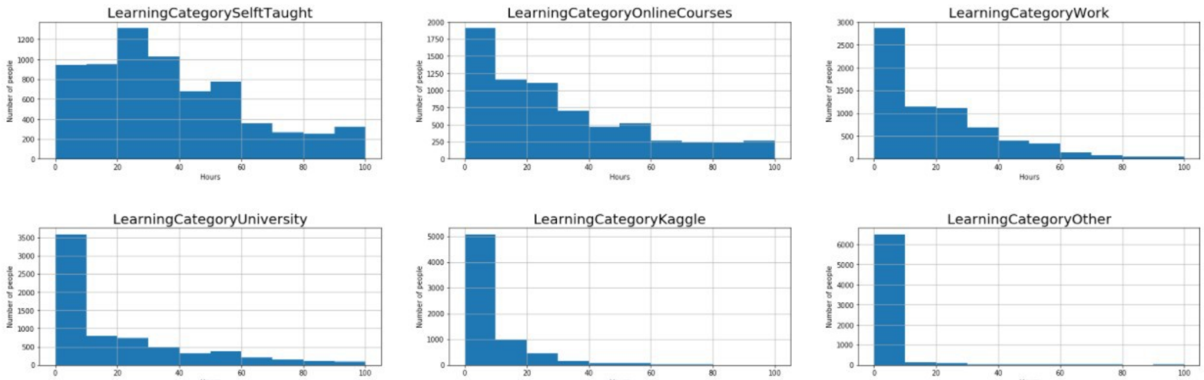
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16716 entries, 0 to 16715
Data columns (total 19 columns):
GenderSelect                     16621 non-null object
Country                          16595 non-null object
Age                              16385 non-null float64
EmploymentStatus                 16716 non-null object
CodeWriter                       13186 non-null object
CurrentJobTitleSelect            11830 non-null object
MLToolNextYearSelect             10998 non-null object
MLMethodNextYearSelect           10833 non-null object
LanguageRecommendationSelect     10998 non-null object
FormalEducation                  15015 non-null object
MajorSelect                      13281 non-null object
Tenure                           13532 non-null object
FirstTrainingSelect              14712 non-null object
LearningCategorySelftTaught      13109 non-null float64
LearningCategoryOnlineCourses    13126 non-null float64
LearningCategoryWork             13111 non-null float64
LearningCategoryUniversity       13122 non-null float64
LearningCategoryKaggle           13126 non-null float64
LearningCategoryOther            13094 non-null float64
dtypes: float64(7), object(12)
memory usage: 2.4+ MB
```

Finding anomalies

```
plt.subplots(figsize=(15,8))
response['Age'].hist(bins=50)
plt.xticks(list(range(0,110,5)))
plt.title('Age Distribution')
plt.show()
```




```
import itertools
Learningtype=
["LearningCategorySelftTaught","LearningCategoryOnlineCourses","LearningCategoryWork","LearningCategoryUniversity","LearningCate
goryKaggle","LearningCategoryOther"]
['WorkToolsFrequencyAmazonML','WorkToolsFrequencyAWS','WorkToolsFrequencyCloudera','WorkToolsFrequencyHadoop','WorkToolsFrequenc
yAzure']
plt.subplots(figsize=(30,20))
length=len(cloud)
for i,j in itertools.zip_longest(Learningtype,range(length)):
    plt.subplot((length/2+1),3,j+1)
    plt.subplots_adjust(wspace=0.2,hspace=0.5)
    response_new[i].hist()
    plt.title(i,size=20)
    plt.ylabel('Number of people')
    plt.xlabel('Hours')
plt.show()
```



Selecting the most appropriate values

```
response = response.loc[(response['Age'] >15) & (response['Age']<85)]
response.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6883 entries, 3 to 16712
Data columns (total 19 columns):
GenderSelect                      6883 non-null object
Country                           6883 non-null object
Age                               6883 non-null float64
EmploymentStatus                  6883 non-null object
CodeWriter                        6883 non-null object
CurrentJobTitleSelect             6883 non-null object
MLToolNextYearSelect              6883 non-null object
MLMethodNextYearSelect            6883 non-null object
LanguageRecommendationSelect      6883 non-null object
FormalEducation                   6883 non-null object
MajorSelect                       6883 non-null object
Tenure                            6883 non-null object
FirstTrainingSelect               6883 non-null object
LearningCategorySelftTaught       6883 non-null float64
LearningCategoryOnlineCourses     6883 non-null float64
LearningCategoryWork              6883 non-null float64
LearningCategoryUniversity        6883 non-null float64
LearningCategoryKaggle            6883 non-null float64
LearningCategoryOther             6883 non-null float64
dtypes: float64(7), object(12)
memory usage: 1.1+ MB
```

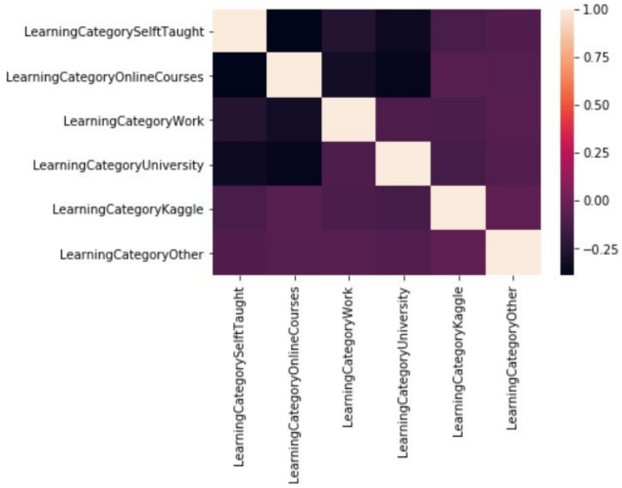
```
response = response.dropna()
response.isnull().sum()
```

```
GenderSelect                      0
Country                           0
Age                               0
EmploymentStatus                  0
CodeWriter                        0
CurrentJobTitleSelect             0
MLToolNextYearSelect              0
MLMethodNextYearSelect            0
LanguageRecommendationSelect      0
FormalEducation                   0
MajorSelect                       0
Tenure                            0
FirstTrainingSelect               0
LearningCategorySelftTaught       0
LearningCategoryOnlineCourses     0
LearningCategoryWork              0
LearningCategoryUniversity        0
LearningCategoryKaggle            0
LearningCategoryOther             0
dtype: int64
```

Finding correlation

```
sns.heatmap(cor,xticklabels=cor.columns,yticklabels=cor.columns)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1743632f160>
```

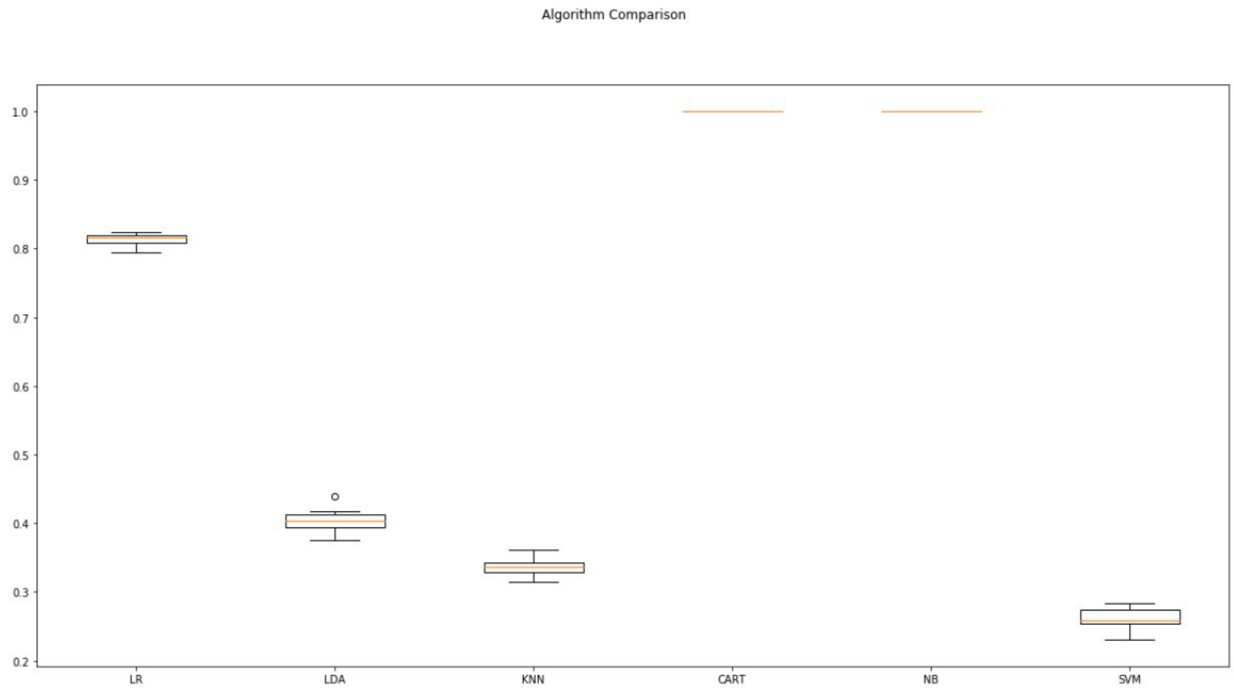


Results

```
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.812568 (0.010174)
LDA: 0.404285 (0.016380)
KNN: 0.336537 (0.014270)
CART: 1.000000 (0.000000)
NB: 1.000000 (0.000000)
SVM: 0.261527 (0.015158)
```

From the above we can conclude that Logistic Regression gives better accuracy for the given data set



To determine the confusion matrix

```
# Make predictions on validation dataset
lr = LogisticRegression()
lr.fit(X_train, Y_train)
predictions = lr.predict(X_validation)
print("accuracy_score")
print(accuracy_score(Y_validation, predictions))
print("confusion_matrix")
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

```
accuracy_score
0.8104575163398693
confusion_matrix
[[314   0   0   0   0   0]
 [  0 351   0   9   0   0]
 [  0 181  23   5   1   0]
 [  0   0   4  18  39   0]
 [  0   0  12  10 150   0]
 [  0   0   0   0   0 260]]
             precision    recall  f1-score   support

        0.0       1.00      1.00      1.00       314
        1.0       0.66      0.97      0.79       360
        2.0       0.59      0.11      0.18       210
        3.0       0.43      0.30      0.35        61
        4.0       0.79      0.87      0.83       172
        5.0       1.00      1.00      1.00       260

avg / total       0.80      0.81      0.77      1377
```

Discussion

From the above experiment we can conclude that there are many types of machine learning techniques which may be appropriate for each circumstance. Without proper EDA and effort to try out different algorithm it is not possible to determine the best solution for any given task.

References

- https://www.kaggle.com/mhajabri/what-do-kagglers-say-about-data-science
- https://www.kaggle.com/ash316/novice-to-grandmaster
- https://www.kaggle.com/rounakbanik/data-science-faq
- https://www.medcalc.org/manual/logistic_regression.php
- https://en.wikipedia.org/wiki/Linear_discriminant_analysis