IST 718 Final Project
Kanning Wu
June 13, 2020

Stack Overflow 2018 Developer Survey Analysis

Introduction
This project topic I chose is 2018 Stack Overflow Developer Survey results from Kaggle. The data is about everything about developers' favorite technologies and job preferences. As a current software engineer, interested in data science, I decided to predict Python and R.

In [1]:

```
import pandas as pd
import numpy as np
from copy import deepcopy

import seaborn as sns

import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import scikitplot as skplt

import hdbscan
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix, precision_score, accuracy_score, make_scorer
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import LeaveOneOut
from sklearn.preprocessing import scale
from sklearn.base import clone

from plotly import tools
#import plotly.plotly as py

import os
from collections import Counter

# TODO
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.figure_factory as ff
# Squarify for treemaps
import squarify
# Random for well, random stuff
import random
# operator for sorting dictionaries
import operator
# For ignoring warnings
import warnings
warnings.filterwarnings('ignore')
```

Dataset

There are two datasets. The first one is surveyresultpublic and the second one is surveyresultschema. The surveyresultpublic contains the main survey results, one respondent per row and one column per question. The surveyresultsschema contains each column name from the main results along with the question text corresponding to that column.

In [2]:
```
df = pd.read_csv('/Users/orange/Downloads/stack-overflow-2018-developer-survey/survey_results_public.csv')
df
```

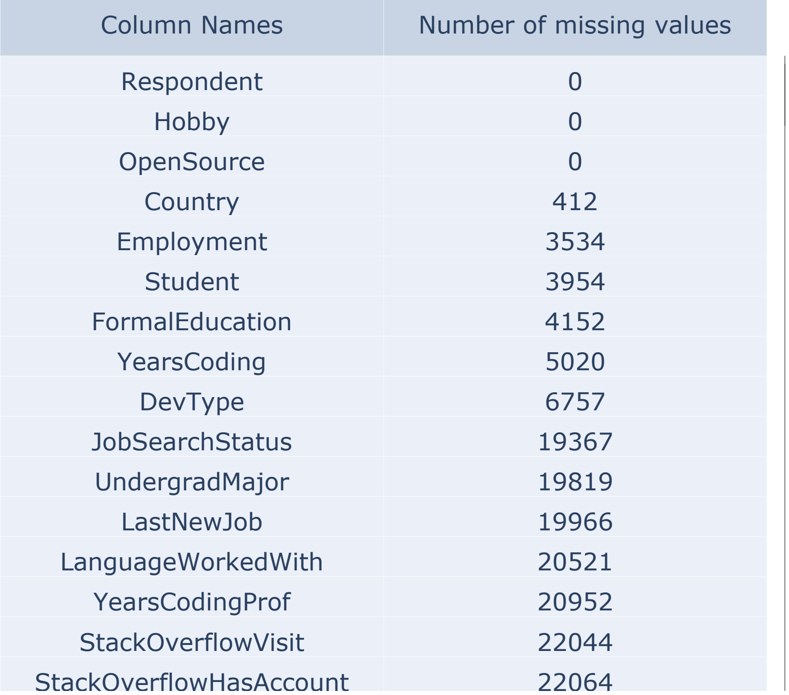
Out[2]:

	Respondent	Hobby	OpenSource	Country	Student	Employment	FormalEducation	Un
0	1	Yes	No	Kenya	No	Employed part-time	Bachelor’s degree (BA, BS, B.Eng., etc.)	M
1	3	Yes	Yes	United Kingdom	No	Employed full-time	Bachelor’s degree (BA, BS, B.Eng., etc.)	A n ch
2	4	Yes	Yes	United States	No	Employed full-time	Associate degree	e
3	5	No	No	United States	No	Employed full-time	Bachelor’s degree (BA, BS, B.Eng., etc.)	e
4	7	Yes	No	South Africa	Yes, part-time	Employed full-time	Some college/university study without earning ...	e
...	...	...	...	...	...	...	...	
98850	101513	Yes	Yes	United States	NaN	NaN	NaN	
98851	101531	No	Yes	Spain	Yes, full-time	Not employed, but looking for work	NaN	
98852	101541	Yes	Yes	India	Yes, full-time	Employed full-time	Bachelor’s degree (BA, BS, B.Eng., etc.)	
98853	101544	Yes	No	Russian Federation	No	Independent contractor, freelancer, or self-em...	Some college/university study without earning ...	
98854	101548	Yes	Yes	Cambodia	NaN	NaN	NaN	

98855 rows × 129 columns

Once the data is loaded, it is the time to find the missing values in the dataset. By using isnull(), a table of missing values in each columns is generated.

```
In [3]: missing = df.isnull().sum().sort_values(ascending = True)
        missing_data = pd.concat([missing], axis = 1, keys = ['Total'])
        missing_data['Columns'] = missing_data.index

        iplot([go.Table(
            header = dict(values = ['Column Names', 'Number of missing values'
        ]),
            cells = dict(values = [missing_data.Columns, missing_data.Total])
           )])
```



Column Names	Number of missing values
Respondent	0
Hobby	0
OpenSource	0
Country	412
Employment	3534
Student	3954
FormalEducation	4152
YearsCoding	5020
DevType	6757
JobSearchStatus	19367
UndergradMajor	19819
LastNewJob	19966
LanguageWorkedWith	20521
YearsCodingProf	20952
StackOverflowVisit	22044
StackOverflowHasAccount	22064

Once the missing values in each columns are found, the data can be cleaned and missing values can be treated. There are a few entries contains multiple values. Split these values.

Since this project is about R and Python usage and prediction in data related developers. Only developers who are data analysts or data scientists are considered.

In [4]:
```
dev = [
    'Data or business analyst',
    'Data scientist or machine learning specialist'
]

multiple = [
    'CommunicationTools','EducationTypes','SelfTaughtTypes','HackathonRe
asons',
    'DatabaseWorkedWith','DatabaseDesireNextYear','PlatformWorkedWith',
    'PlatformDesireNextYear','Methodology','VersionControl',
    'AdBlockerReasons','AdsActions','ErgonomicDevices','Gender',
    'SexualOrientation','RaceEthnicity', 'LanguageWorkedWith'
]

df = df.loc[df.DevType.str.contains('|'. join(dev)).fillna(False)]

df.drop([
    'IDE', 'FrameworkWorkedWith', 'FrameworkDesireNextYear',
    'LanguageDesireNextYear', 'DevType', 'CurrencySymbol',
    'Salary', 'SalaryType', 'Respondent', 'Currency'
], axis = 1, inplace = True)

for c in multiple:

    temp = df[c].str.split(';', expand=True)

    new_columns = pd.unique(temp.values.ravel())
    for new_c in new_columns:
        if new_c and new_c is not np.nan:

            idx = df[c].str.contains(new_c, regex=False).fillna(False)
            df.loc[idx, f"{c}_{new_c}"] = 1

    df.drop(c, axis=1, inplace=True)


df = pd.get_dummies(df)
```

After clean the columns, find the columns with dummy variables and put zeros for missing values. Columns don't contain dummy variables are treated with median value for NAs

In [5]:
```
df.dropna(axis=1, how='all', inplace=True)

# Find dummies
dummies = [c for c in df.columns if len(df[c].unique()) == 2]
non_dummies = [c for c in df.columns if c not in dummies]

df[dummies] = df[dummies].fillna(0)
df[non_dummies] = df[non_dummies].fillna(df[non_dummies].median())
```

The dataset has several columns which may have high collinearity. This may affect the analysis accuracy.

In [6]:

```
corr_matrix = df.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
to_drop = [column for column in upper.columns if any(upper[column] > 0.70)]
df = df.drop(to_drop, axis=1)
```

Once dataset are cleaned, it is the time to split X and y variables. Only R and Python are considered in this project. All data with other languages are dropped. Among these R and/or Python users, they are separated into 3 groups. 1. only Python users, 2. only R users and 3. both R and Python users.

In [7]:

```
df = df[df.ConvertedSalary < df.ConvertedSalary.mean() + df.ConvertedSalary.std()*3]


print(df[c] for c in df.columns if df[c])

#All dummies in this dataset are less or equal to 1
nondummy_columns = [c for c in df.columns if df[c].max() > 1]
scaled_df = deepcopy(df)
scaled_df.loc[:, nondummy_columns] = scale(df[nondummy_columns])

R_only = (df.LanguageWorkedWith_R == 1) & (df.LanguageWorkedWith_Python == 0)
Python_only = (df.LanguageWorkedWith_R == 0) & (df.LanguageWorkedWith_Python == 1)
R_and_Python = (df.LanguageWorkedWith_R == 1) & (df.LanguageWorkedWith_Python == 1)

scaled_df.loc[R_only, 'R_or_Python'] = 0
scaled_df.loc[Python_only, 'R_or_Python'] = 1
scaled_df.loc[R_and_Python, 'R_or_Python'] = 2
scaled_df.dropna(subset=['R_or_Python'], axis=0, inplace=True)

y_all = scaled_df['R_or_Python']
X_all = scaled_df.drop(['LanguageWorkedWith_Python', 'LanguageWorkedWith_R', 'R_or_Python'], axis=1)

df_only = scaled_df[scaled_df.R_or_Python != 2]
y_only = df_only['R_or_Python']
X_only = df_only.drop(['LanguageWorkedWith_Python', 'LanguageWorkedWith_R', 'R_or_Python'], axis=1)

prefixes = ['LanguageWorkWith_', 'PlatformWorkedWith_', 'DatabaseWorkedWith_', 'OperatingSystem_']
drop2 = [c for c in X_only.columns if any(check in c for check in prefixes)]

df.drop(drop2, axis = 1, inplace = True)
```

```
<generator object <genexpr> at 0x7fe0b35d4dd0>
```

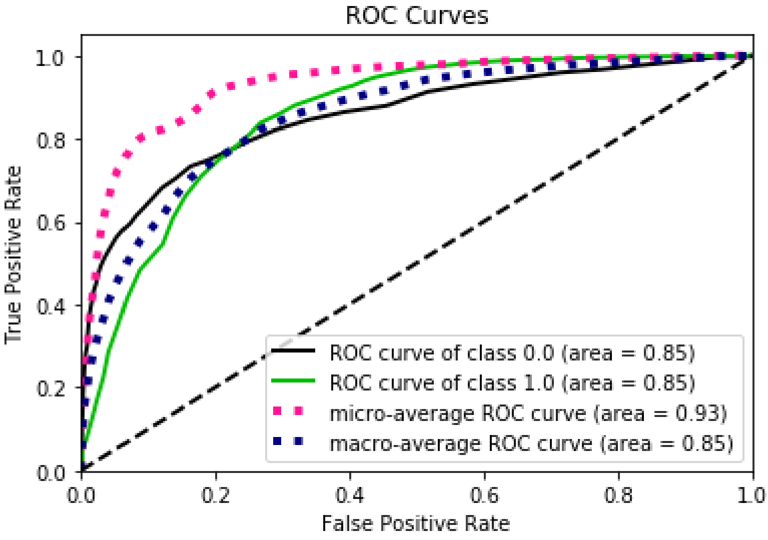
Analysis After all the steps above, the first analysis can get started by using Random Forest Classifier

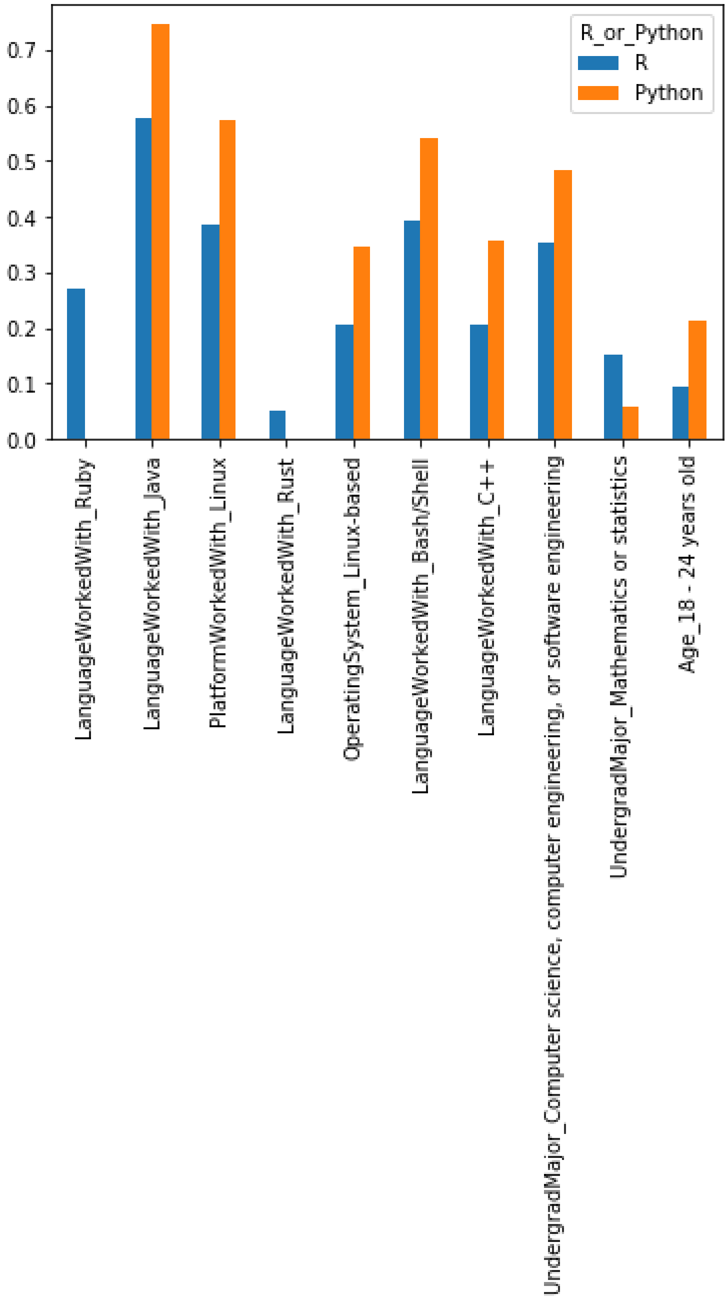
In [8]:

```
rfc = ExtraTreesClassifier(n_jobs = -1, n_estimators = 100, class_weight
= 'balanced')

rfc_pred = rfc.fit(X_only, y_only)

predicted = cross_val_predict(
    clone(rfc),
    X_only,
    y_only,
    cv = 20,
    n_jobs = -1,
    verbose = 0,
    method = 'predict_proba'
)
```

In [9]:

```
skplt.metrics.plot_roc(y_only, predicted)


if hasattr(rfc, 'feature_importances_'):

        importances = rfc.feature_importances_
        index = np.argsort(importances)[::-1][:10]
else:

        importances = rfc.coef_[0]
        index = np.argsort(np.abs(rfc.coef_[0]))[::-1][:10]


X_only[X_only.columns[index]].\
    groupby(y_only).mean().T. \
    rename(columns = {0.0: "R", 1.0: "Python"}).\
    plot(kind = 'bar')
plt.show()
```


ROC Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 0.0 (area = 0.85)
ROC curve of class 1.0 (area = 0.85)
micro-average ROC curve (area = 0.93)
macro-average ROC curve (area = 0.85)

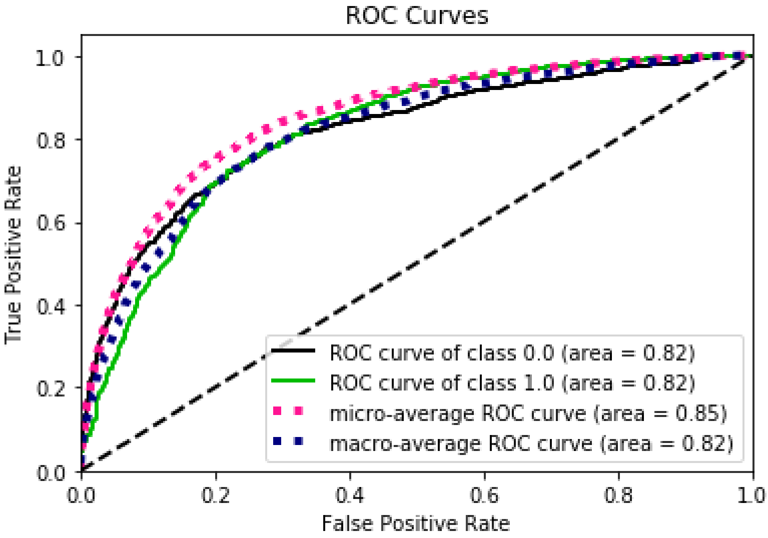

R_or_Python
R
Python
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
LanguageWorkedWith_Ruby
LanguageWorkedWith_Java
PlatformWorkedWith_Linux
LanguageWorkedWith_Rust
OperatingSystem_Linux-based
LanguageWorkedWith_Bash/Shell
LanguageWorkedWith_C++
UndergradMajor_Computer science, computer engineering, or software engineering
UndergradMajor_Mathematics or statistics
Age_18 - 24 years old

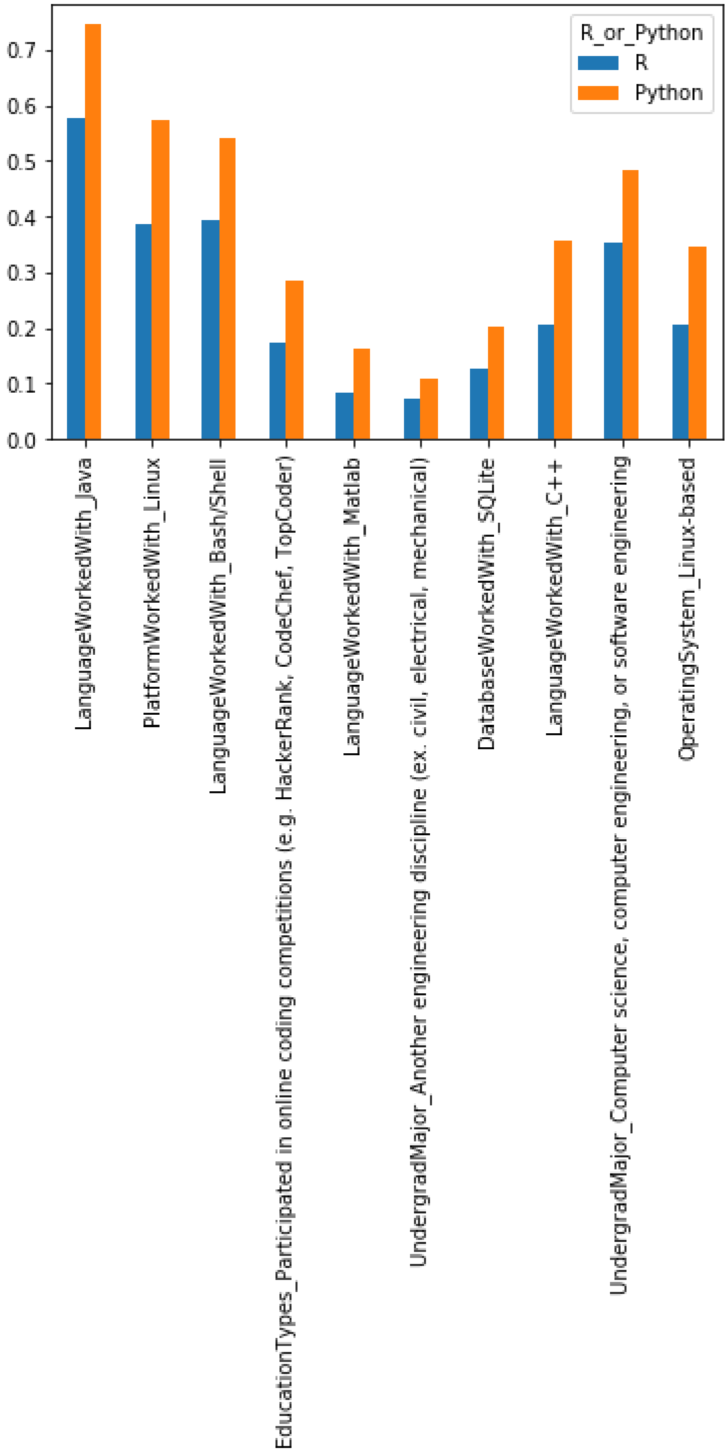
In [10]:

```
# Logistic Regression
lr = LogisticRegression(class_weight='balanced', C=0.01)

lr_pred = lr.fit(X_only, y_only)

predicted = cross_val_predict(
    clone(lr),
    X_only,
    y_only,
    cv = 20,
    n_jobs = -1,
    verbose = 0,
    method = 'predict_proba'
)
```

In [11]:

```
skplt.metrics.plot_roc(y_only, predicted)

importances = lr.coef_[0]
index = np.argsort(importances)[::-1][:10]

X_only[X_only.columns[index]].\
    groupby(y_only).mean().T. \
    rename(columns = {0.0: "R", 1.0: "Python"}).\
    plot(kind = 'bar')
plt.show()
```


ROC Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 0.0 (area = 0.82)
ROC curve of class 1.0 (area = 0.82)
micro-average ROC curve (area = 0.85)
macro-average ROC curve (area = 0.82)


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
R_or_Python
R
Python
LanguageWorkedWith_Java
PlatformWorkedWith_Linux
LanguageWorkedWith_Bash/Shell
EducationTypes_Participated in online coding competitions (e.g. HackerRank, CodeChef, TopCoder)
LanguageWorkedWith_Matlab
UndergradMajor_Another engineering discipline (ex. civil, electrical, mechanical)
DatabaseWorkedWith_SQLite
LanguageWorkedWith_C++
UndergradMajor_Computer science, computer engineering, or software engineering
OperatingSystem_Linux-based

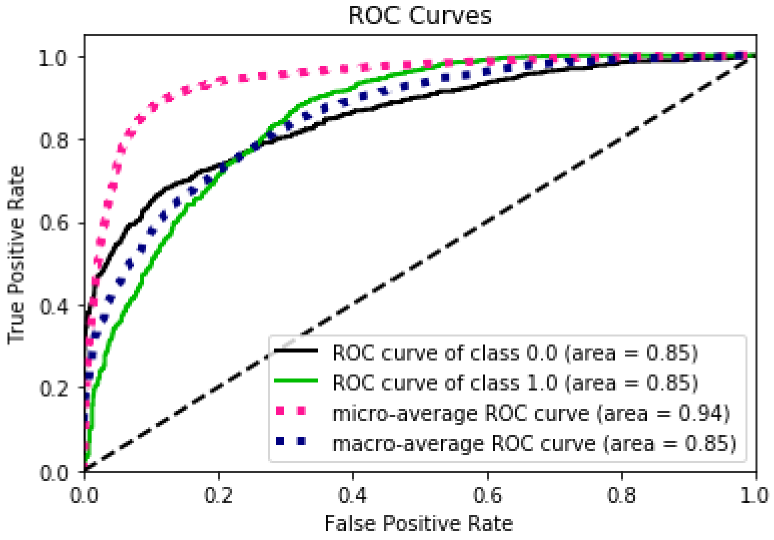
In [12]:

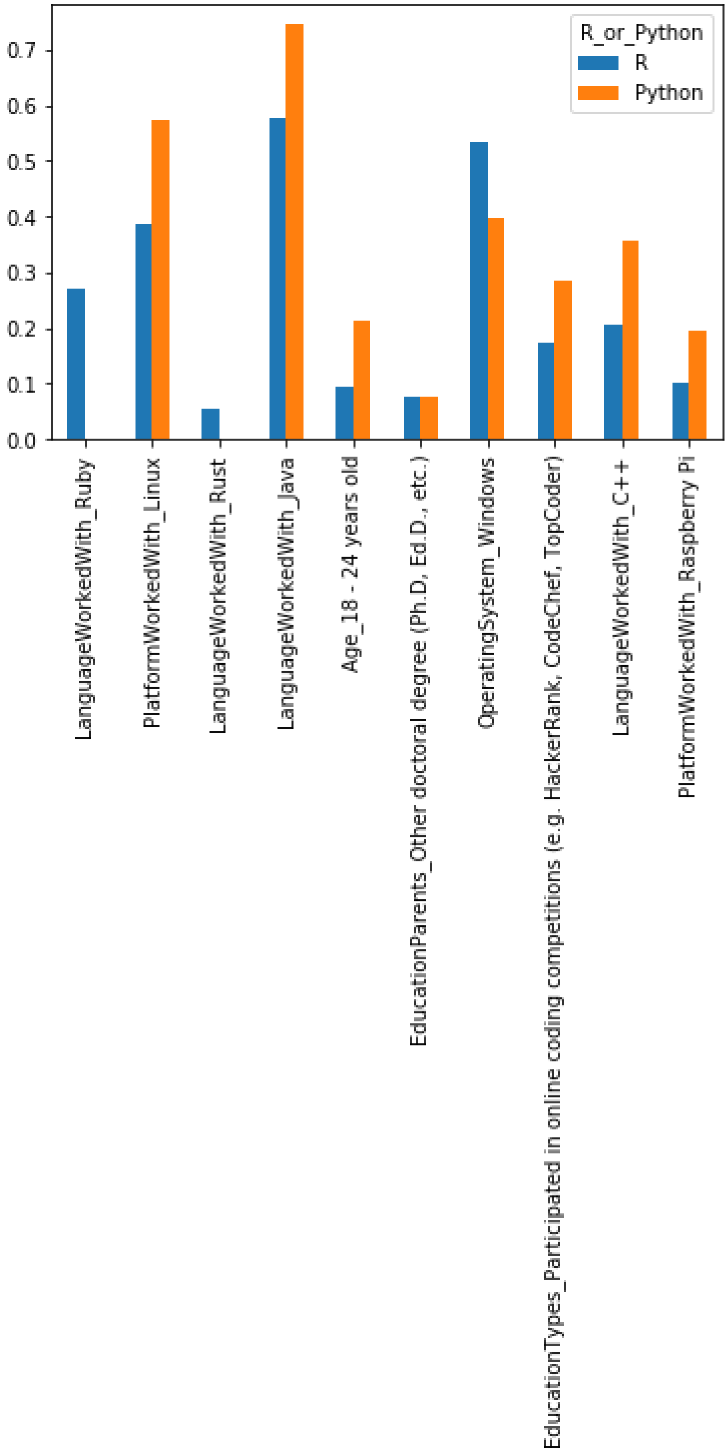
```
xgb = XGBClassifier(n_jobs = -1, n_estimator = 50)

xgb_pred = xgb.fit(X_only, y_only)

predicted = cross_val_predict(
    clone(xgb),
    X_only,
    y_only,
    cv = 20,
    n_jobs = -1,
    verbose = 0,
    method = 'predict_proba'
)
```

```
[10:17:38] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:48
0:
Parameters: { n_estimator } might not be used.

  This may not be accurate due to some parameters are only used in lang
uage bindings but
  passed down to XGBoost core.  Or some parameters are not used but sli
p through this
  verification. Please open an issue if you find above cases.
```

In [13]:

```
skplt.metrics.plot_roc(y_only, predicted)

importances = xgb.feature_importances_
index = np.argsort(importances)[::-1][:10]

X_only[X_only.columns[index]]. \
    groupby(y_only).mean().T. \
    rename(columns = {0.0: "R", 1.0: "Python"}).\
    plot(kind = 'bar')
plt.show()
```


ROC Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 0.0 (area = 0.85)
ROC curve of class 1.0 (area = 0.85)
micro-average ROC curve (area = 0.94)
macro-average ROC curve (area = 0.85)


R_or_Python
R
Python
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
LanguageWorkedWith_Ruby
PlatformWorkedWith_Linux
LanguageWorkedWith_Rust
LanguageWorkedWith_Java
Age_18 - 24 years old
EducationParents_Other doctoral degree (Ph.D, Ed.D., etc.)
OperatingSystem_Windows
EducationTypes_Participated in online coding competitions (e.g. HackerRank, CodeChef, TopCoder)
LanguageWorkedWith_C++
PlatformWorkedWith_Raspberry Pi

From above results,

Random Forest Classifier has a ROC of 0.86 and predicted that

```
1. If you are a Ruby user, you will be a R user.
2. If you are a Rust user, you will be a R user.
3. If you have a undergraduate degree of math or stats, you will be a R user
4. If you are a Linux user, you will be a Python user.
5. If you are a Java user, you will be a Python user.
6. If you are a C++ user, you will be a Python user.
7. If you use bash/shell, you will be a Python user.
8. If you have a undergraduate degree of computer science or software engineering, you will be a Python user.
9. If you are between age of 18-24, you will be a Python user.
```

Logistic Regression has a ROC of 0.82 and predicted that you will be a Python user if you are a user of Java, Linux, bash/shell, SQLite, Matlab, C++ and participated in a competition.

XGBosst has a Roc of 0.85 and predicted tht

```
1. If you are a Ruby user, you will be a R user.
2. If you are a Windows user, you will be a R user.
3. If you are a Rust user, you will be a R user.
4. If you have a Doctoral, you will have a 50% chance of being a R user and
 50% chance of being a Python user.
5. If you are a Linux user, you will be a Python user.
6. If you are a Java user, you will be Python user
7. If you participated an competition, you will be a Python user
8. If you are a C++ user, you will be a Python user
9. If you are a Raspberry Pi user, you will be a Python  user.
10 If you are between age of 18-24, you will be a Python user
```

All of these three algorithms have high accuracy, based on their ROC values. Also, there is no conflicts among the results generated by these three algorithms.

Reference https://www.kaggle.com/stackoverflow/stack-overflow-2018-developer-survey (https://www.kaggle.com/stackoverflow/stack-overflow-2018-developer-survey) https://www.kaggle.com/nanomathias/ (https://www.kaggle.com/nanomathias/)

```
In [ ]:
```