In [1]:
```
# data base
import pymysql
from sqlalchemy import create_engine
import MySQLdb


import numpy as np
import pandas as pd
import math
%matplotlib inline
from matplotlib import pyplot as plt
import seaborn as sns
from collections import Counter
import scipy.stats as ss
from dython import nominal
import ipywidgets as widgets
from ipywidgets import interact, interact_manual
import re
import string
import nltk
from nltk.corpus import stopwords
#nltk.download('stopwords')
from nltk.tokenize import RegexpTokenizer
from nltk.stem import PorterStemmer

#tf-idf libraries
from sklearn.feature_extraction.text import TfidfVectorizer

#importing truncated svd for LSA- Latent Semantic Analysis
from sklearn.decomposition import TruncatedSVD

#Importing tsne
from sklearn.manifold import TSNE

#Importing MiniBatch k-means
from sklearn.cluster import MiniBatchKMeans

#Silhouette score for clusters
from sklearn.metrics import silhouette_score

#Importing Kmeans
from sklearn.cluster import KMeans

#Importing bokeh

from bokeh.plotting import figure
from bokeh.io import show, output_notebook
from bokeh.models import HoverTool
from bokeh.palettes import Spectral6
from bokeh.models import ColumnDataSource
```

In [3]:
```
#Open database connection"
engine = create_engine('mysql+pymysql://root:Root@123@localhost/gitclass',echo=False)
```

In [5]:
```
df=pd.read_sql("Select * from gitinfo_cleaned",con=engine)
df.shape
```

Out[5]:
```
(514640, 14)
```

In [6]:
```
df_repo=df[['repository_name','repository_description','repository_language']]
```

In [7]:
```
df_repo.repository_language.unique()
```

Out[7]:
```
array(['JavaScript', 'CSS', 'C', 'other', 'Ruby', 'Python', 'Java', 'PHP'],
      dtype=object)
```

In [8]:
```
# Finding lenth of the text
Length_of_Text=df_repo.repository_description.apply(len)
```

In [9]:
```
Length_of_Text.mean()
```

Out[9]:
```
50.99888854344785
```

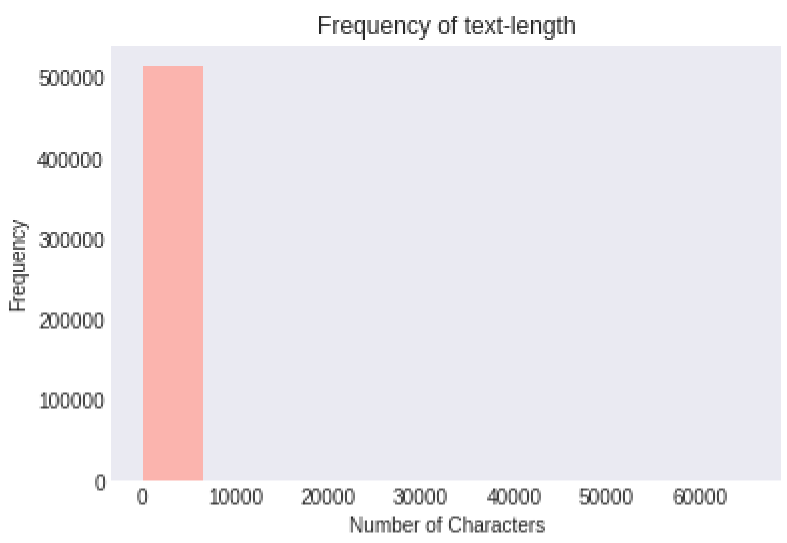
In [10]:
```
Length_of_Text.median()
```

Out[10]:
```
40.0
```

In [11]:
```
Length_of_Text.describe()
```

Out[11]:
```
count    514640.000000
mean         50.998889
std         267.612682
min           1.000000
25%          25.000000
50%          40.000000
75%          61.000000
max       65535.000000
Name: repository_description, dtype: float64
```

In [12]:
```
plt.style.use('seaborn-dark')
Length_of_Text.plot(kind='hist',bins=10,colormap='Pastel1');
plt.xlabel("Number of Characters");
plt.title('Frequency of text-length');
```



In [13]:
```
# Function to do text cleaning and give tokens
def text_cleaning_tokens(text):
    stop = set(stopwords.words('english'))
    text = text.lower()
    regex = re.compile('[' +re.escape(string.punctuation) + '0-9\\r\\t\\n]')
    text = regex.sub(" ", text) # remove punctuation
    #stripping- removes any white space characters
    text=text.strip()
    tokenizer = RegexpTokenizer("[\w']+")
    tokens=tokenizer.tokenize(text)
    filtered_tokens= [word for word in tokens if not word in stop]
    filtered_tokens= [w.lower() for w in filtered_tokens]
    return filtered_tokens
```

In [14]:
```
# Function to do only text cleaning
#Text Cleaning function
def text_cleaning(text):
    stop = set(stopwords.words('english'))
    text = text.lower()
    regex = re.compile('[' +re.escape(string.punctuation) + '0-9\\r\\t\\n]')
    text = regex.sub(" ", text) # remove punctuation
    #stripping
    text=text.strip()
    # removes one letter words and numbers
    pattern=re.compile(r'\W*\b\w{1,3}\b')
    text=pattern.sub('', text)
    #Removing non-english words
    text = re.sub("([^\x00-\x7F])+"," ",text)
    return text
```

In [15]:
```
df_repo['cleaned_text']=df_repo.repository_description.apply(text_cleaning)
```

```
/home/isiia/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  """Entry point for launching an IPython kernel.
```

In [16]:
```
df_repo['tokens']=df_repo.repository_description.apply(text_cleaning_tokens)
```

```
/home/isiia/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  """Entry point for launching an IPython kernel.
```

In [17]:
```python
df_repo[['repository_description','tokens']].head()
```

Out[17]:

	repository_description	tokens
0	CSS3 fun. Box shadows, opacity, border-radius,...	[css, fun, box, shadows, opacity, border, radi...
1	A OSX style dock using CSS and a few lines of ...	[osx, style, dock, using, css, lines, jquery]
2	Read repeated input from a file or stdin.	[read, repeated, input, file, stdin]
3	qstoq item form app	[qstoq, item, form, app]
4	Learn about media queries and responsive design	[learn, media, queries, responsive, design]

In [18]:
```python
for description, tokens in zip(df_repo['repository_description'].head(),
                               df_repo['tokens'].head()):
    print('description:', description)
    print('tokens:', tokens)
    print()
```

```
description: CSS3 fun. Box shadows, opacity, border-radius, and transitions
tokens: ['css', 'fun', 'box', 'shadows', 'opacity', 'border', 'radius', 'transitions']

description: A OSX style dock using CSS and a few lines of jQuery
tokens: ['osx', 'style', 'dock', 'using', 'css', 'lines', 'jquery']

description: Read repeated input from a file or stdin.
tokens: ['read', 'repeated', 'input', 'file', 'stdin']

description: qstoq item form app
tokens: ['qstoq', 'item', 'form', 'app']

description: Learn about media queries and responsive design
tokens: ['learn', 'media', 'queries', 'responsive', 'design']
```

Exploratory Data Analysis for pre-processed text-WordCloud

In [19]:
```python
# Import the wordcloud library
from wordcloud import WordCloud
# Join the different processed titles together.
long_string = ''.join(df_repo['repository_description'].values)
# Create a WordCloud object
wordcloud = WordCloud(background_color="white", max_words=5000, contour_width=3, contour_color='steelblue')
# Generate a word cloud
wordcloud.generate_from_text(long_string)
# Visualize the word cloud
#
# plt.axis("off")
# plt.show()
# wordcloud.to_image();
# wordcloud.to_file("/Users/vinatigattupalli/Github_Classification/Visualizations/wordcloud.png")
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud,interpolation='bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

In [20]:
```
plt.savefig("/home/isiia/Github_Classification/Data_Visualization/wordcloud.png", facecolor='k', bbox_inches='ti
```

```
<Figure size 432x288 with 0 Axes>
```

In [21]:
```
# getting the word clouds category wise
lan_repo_dict=dict()
df_repo.repository_language.astype('category').head()
```

Out[21]:
```
0    JavaScript
1           CSS
2             C
3           CSS
4    JavaScript
Name: repository_language, dtype: category
Categories (8, object): [C, CSS, Java, JavaScript, PHP, Python, Ruby, other]
```

Term Frequency- Inverse term frequency(TF-IDF)

In [22]:
```
tokenizer = RegexpTokenizer("[\w']+")
vectorizer = TfidfVectorizer(min_df=10,
                             max_features=5000,
                             tokenizer=tokenizer.tokenize,
                             ngram_range=(1, 2),stop_words='english',analyzer='word')
```

In [23]:
```
#Fitting tf-idf on repository descriptions

all_desc = df_repo['cleaned_text'].values
vz = vectorizer.fit_transform(list(all_desc))
print(vz.shape)
# getting the tf-idf values for the words in repository descrition #1
print(vz[0])
```

```
(514640, 5000)
  (0, 4594)     0.5347215480046913
  (0, 3494)     0.591109309134944
  (0, 461)      0.6038730419164907
```

In [24]:
```
vz.shape
```

Out[24]:
```
(514640, 5000)
```

In [27]:
```
# vz is a matrix with tfidf values for all the words with respect to each document or words cluster.
vectorizer.get_feature_names()
```

Out[27]:
```
['ability',
 'able',
 'absolute',
 'abstract',
 'abstraction',
 'abstraction layer',
 'academic',
 'academy',
 'accelerated',
 'accelerometer',
 'accept',
 'acceptance',
 'access',
 'access contributing',
 'access control',
 'accessibility',
 'accessible',
 'accessing',
 'accompanying',
```

The shape of the our repository_description after doing TfidfVectorier was (514640, 5000).

So we had 514640 repository_description in the dataset and each review is vectorized having dimension of 5000.

[https://stackoverflow.com/questions/46959801/understanding-the-matrix-output-of-tfidfvectorizer-in-sklearn (https://stackoverflow.com/questions/46959801/understanding-the-matrix-output-of-tfidfvectorizer-in-sklearn)](https://stackoverflow.com/questions/46959801/understanding-the-matrix-output-of-tfidfvectorizer-in-sklearn)

Understanding Tfidfvectorizer- [https://kavita-ganesan.com/tfidftransformer-tfidfvectorizer-usage-differences/#.XqHeeFNKhQI (https://kavita-ganesan.com/tfidftransformer-tfidfvectorizer-usage-differences/#.XqHeeFNKhQI)](https://kavita-ganesan.com/tfidftransformer-tfidfvectorizer-usage-differences/#.XqHeeFNKhQI)

tf-idf now is a the right measure to evaluate how important a word is to a document in a collection or corpus.

In [28]:
```
#  create a dictionary mapping the tokens to their tfidf values
tfidf = dict(zip(vectorizer.get_feature_names(), vectorizer.idf_))

#create a data frame of the tokens and their tfidf values
tfidf = pd.DataFrame(columns=['tfidf']).from_dict(
                    dict(tfidf), orient='index')
tfidf.columns = ['tfidf']
```

In [29]:
```
tfidf.sort_values(by=['tfidf'], ascending=True).head(10)
```

Out[29]:

	tfidf
project	4.140588
plugin	4.222070
simple	4.252197
using	4.313824
library	4.346840
code	4.405327
based	4.518497
data	4.616846
application	4.673992
python	4.723357

TFIDF is the product of the TF and IDF scores of the term. Higher the TFIDF score, the rarer the term is and vice-versa.

Applying truncatedSVD to use it reduce the dimensions

In [30]:
```
#Applying LSA on the tf-idf vector
n_comp = 2000
#defining truncated SVD
svd_obj = TruncatedSVD(n_components=n_comp, algorithm='arpack')

#fitting the tf-idf values into the truncatedsvd
svd_response=svd_obj.fit_transform(vz)
```

The fraction of variance explained by a principal component is the ratio between the variance of that principal component and the total variance. For several principal components, add up their variances and divide by the total variance.

In [31]:
```
var_explained = svd_obj.explained_variance_ratio_.sum()
var_explained
```

Out[31]:
```
0.8083222499963875
```

In [32]:
```
svd_response
```

Out[32]:
```
array([[ 1.36465125e-04,  2.10580920e-04,  2.85625526e-04, ...,
        -6.34155439e-05,  9.24365119e-04,  4.69601046e-03],
       [ 1.83898071e-02,  9.44222484e-03,  1.30863894e-02, ...,
        -2.66201965e-03, -1.41090002e-03, -1.32344732e-03],
       [ 5.19956023e-03,  3.75856404e-03,  6.09960273e-03, ...,
         1.12448169e-04,  5.88788312e-04, -1.32725112e-04],
       ...,
       [ 2.23647978e-03,  3.94345344e-03,  5.88051118e-03, ...,
         5.28353069e-04, -2.95046550e-03, -1.75110768e-03],
       [ 3.81008426e-03,  1.08549729e-02,  1.82788257e-02, ...,
         1.21375310e-03,  2.11970437e-03,  6.67188350e-04],
       [ 3.33340862e-03,  8.51520245e-03,  1.20158087e-02, ...,
        -2.00177014e-03,  2.76468698e-03, -3.77179228e-03]])
```

In [33]:
```
df_svd = pd.DataFrame(svd_response)
```

In [34]:
```
df_svd.columns = ['svd_item_'+str(i) for i in range(n_comp)]
```

In [35]:
```
df_svd.head()
```

Out[35]:

	svd_item_0	svd_item_1	svd_item_2	svd_item_3	svd_item_4	svd_item_5	svd_item_6	svd_item_7	svd_item_8	svd_item_9	...	svd_item_1990	svd_it
0	0.000136	0.000211	0.000286	0.000220	0.000416	0.001000	-0.000493	-0.000376	-0.000317	-0.000024	...	0.000009	-
1	0.018390	0.009442	0.013086	0.012371	0.035012	0.063249	-0.032237	-0.030743	-0.016786	0.015564	...	-0.001394	
2	0.005200	0.003759	0.006100	0.004448	0.012594	0.027181	-0.010500	-0.010093	-0.006239	0.002955	...	0.001327	
3	0.001821	0.001352	0.001487	0.001218	0.003073	0.006816	-0.003648	-0.003453	-0.001525	0.000837	...	-0.004843	-
4	0.002364	0.005420	0.005228	0.003094	0.006620	0.012824	-0.002898	-0.005748	0.000191	0.004156	...	0.002140	-

5 rows × 2000 columns

Plotting the tfidf matrix

1. We will be taking a small sample to plot the tfidf matrix.

2. Apply Truncated SVD on the sample and reduce its dimensions.

3. Apply t-sne to reduce the dimensions to 2 or 3 easier for visualization

Given the high dimension of our tfidf matrix,we will be using a sample and we need to reduce their dimension using the Singular Value Decomposition (SVD) technique. And to visualize our vocabulary, we could next use t-SNE to reduce the dimension from 50 to 2. t-SNE is more suitable for dimensionality reduction to 2 or 3.

In [36]:
```
# Getting the sample from our data
samp_size= 20000

#getting a stratified sampling function
def stratified_sample_df(df, col, n_samples):
    n = min(n_samples, df[col].value_counts().min())
    df_ = df.groupby(col).apply(lambda x: x.sample(n,random_state=7))
    df_.index = df_.index.droplevel(0)
    return df_


df_repo_sample=stratified_sample_df(df_repo,'repository_language',samp_size)
```

In [37]:
```
vectorizer_sample = TfidfVectorizer(min_df=10,
                             max_features=300,
                             tokenizer=tokenizer.tokenize,
                             ngram_range=(1, 1),stop_words='english',analyzer='word')
```

In [38]:
```
#applying tfidfvectorizer
vz_sample = vectorizer_sample.fit_transform(list(df_repo_sample['cleaned_text']))

#applying truncatedsvd on vz_sample
n_comp=20
svd_obj_sample = TruncatedSVD(n_components=n_comp, algorithm='arpack')
vz_sample_svd=svd_obj_sample.fit_transform(vz_sample)
```

Difference between explained variance and explained variance ratio

The percentage of the explained variance is:

explained_variance_ratio_

The variance i.e. the eigenvalues of the covariance matrix is:

explained_variance_

In [39]:
```
variance=svd_obj_sample.explained_variance_ratio_
print(variance.sum()*100)
```

```
26.075578727785203
```

In [32]:
```
# #We are trying to reduce the dimensions to 2 using t-sne

tsne_model = TSNE(n_components=2, verbose=1, random_state=7, n_iter=500)
```

In [33]:
```
tsne_sample_ifidf = tsne_model.fit_transform(vz_sample_svd)
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 160000 samples in 1.742s...
[t-SNE] Computed neighbors for 160000 samples in 91.581s...
[t-SNE] Computed conditional probabilities for sample 1000 / 160000
[t-SNE] Computed conditional probabilities for sample 2000 / 160000
[t-SNE] Computed conditional probabilities for sample 3000 / 160000
[t-SNE] Computed conditional probabilities for sample 4000 / 160000
[t-SNE] Computed conditional probabilities for sample 5000 / 160000
[t-SNE] Computed conditional probabilities for sample 6000 / 160000
[t-SNE] Computed conditional probabilities for sample 7000 / 160000
[t-SNE] Computed conditional probabilities for sample 8000 / 160000
[t-SNE] Computed conditional probabilities for sample 9000 / 160000
[t-SNE] Computed conditional probabilities for sample 10000 / 160000
[t-SNE] Computed conditional probabilities for sample 11000 / 160000
[t-SNE] Computed conditional probabilities for sample 12000 / 160000
[t-SNE] Computed conditional probabilities for sample 13000 / 160000
[t-SNE] Computed conditional probabilities for sample 14000 / 160000
[t-SNE] Computed conditional probabilities for sample 15000 / 160000
[t-SNE] Computed conditional probabilities for sample 16000 / 160000
```

In [34]:
```
tsne_df=pd.DataFrame(tsne_sample_ifidf)
tsne_df.columns=['tsne_1','tsne_2']
tsne_df['repository_description'] = df_repo['cleaned_text']
tsne_df['tokens'] = df_repo['tokens']
tsne_df['repository_language'] = df_repo['repository_language']
```

Plotting the tfidf matrix with bokeh

In [35]:
```
output_notebook()
# source = ColumnDataSource(data=dict(x=tsne_df['tsne_1'], y=tsne_df['tsne_2'],
#                                     description=tsne_df['repository_description'],
#                                     repository_language=tsne_df['repository_language'],
#                                  tokens=tsne_df['tokens']))

plot_tfidf = figure(plot_width=700, plot_height=600,
                       title="tf-idf clustering of the repository description",
    tools="pan,wheel_zoom,box_zoom,reset,hover,previewsave",
    x_axis_type=None, y_axis_type=None, min_border=1)

plot_tfidf.scatter(x='tsne_1',y='tsne_2',color='red',source=tsne_df)
#adding interactivity
hover = plot_tfidf.select(dict(type=HoverTool))
hover.tooltips={"description": "@repository_description", "tokens": "@tokens", "repo_language":"@repository_langu
show(plot_tfidf)
```

(https://bokeh.pydata.org) Loading BokehJS ...

Implementing K-means on TF-IDF data

Implementation of mini batch K means

In [137]:
```
# I am selecting the number of clusters as 18 just a feeling because the number of target variables is 18 initia
#Now after using elbow method we decide the number of clusters are 7
num_clusters = 5 # need to be selected wisely

# initializing the mini batch Kmeans model
kmeans_model = MiniBatchKMeans(n_clusters=num_clusters,
                               init='k-means++',
                               n_init=1,
                               init_size=1000, batch_size=1000, verbose=0, max_iter=1000)
```

In [138]:
```
#Fitting the model
kmeans_fit = kmeans_model.fit(vz)
print(kmeans_fit)
```

```
MiniBatchKMeans(batch_size=1000, compute_labels=True, init='k-means++',
                init_size=1000, max_iter=1000, max_no_improvement=10,
                n_clusters=5, n_init=1, random_state=None,
                reassignment_ratio=0.01, tol=0.0, verbose=0)
```

In [139]:
```
# Getting the transformed matrix after fitting
kmeans_distances = kmeans_model.transform(vz)
print(kmeans_distances[0])
print(kmeans_distances)
```

```
[1.00158595 1.04051811 1.223863   1.04283959 1.08466142]
[[1.00158595 1.04051811 1.223863   1.04283959 1.08466142]
 [0.99590874 1.03066275 1.22320491 1.03736651 1.081775  ]
 [0.9983674  1.03628443 1.22347648 1.03872392 1.08338194]
 ...
 [0.99877992 1.03711114 1.22373042 1.0397186  1.08357475]
 [0.9959174  1.03362938 1.2236037  1.03770088 1.07989854]
 [0.99598078 1.03269672 1.22293713 1.03556655 1.08167539]]
```

In [140]:
```
#Getting the centroids of the clusters
centroids=kmeans_fit.cluster_centers_
print(centroids.shape)
print(centroids[0,:-10])

# we have 7 clusters and 7 centroid clusters so each centroid will have 17294 co-ordinates
```

```
(5, 5000)
[2.21609315e-04 2.20104021e-04 6.28122390e-05 ... 7.30015559e-05
 1.96607945e-04 4.78041824e-04]
```

In [141]:
```
# Getting the indices of centroids in descending order
# Argsort is used to sort the indices of the given array and [:,::-1] is used to reverse the list
order_centroids = kmeans_fit.cluster_centers_.argsort()[:, ::-1]
print(order_centroids)

print(order_centroids.shape)
```

```
[[3170 4871 2407 ... 4047 4014 1703]
 [4010 4720 2407 ...  794 1802 3830]
 [ 423 3090 3089 ... 3242 3243    0]
 [ 346 4010 1745 ... 4627 4366 2416]
 [3324 1615 1616 ... 3177 1443 2499]]
(5, 5000)
```

In [142]:
```
# Getting the features from the tf-idf matrix
terms = vectorizer.get_feature_names()
tfidf= vectorizer.idf_
#print(len(terms))
```

In [151]:
```
# printing the terms and their respective clusters
for i in range(0,num_clusters):
    print('\n')
    print("Cluster {} :".format(i))
    cluster_words=''
    for j in order_centroids[i,:50]:
        cluster_words=cluster_words+'|'+terms[j]
    print(cluster_words)

#We printed the terms corresponding to our centroid points
```

```
Cluster 0 :
|plugin|website|library|code|test|android|using|python|data|application|framework|repository|wordpress|game|si
te|client|github|files|mirror|java|javascript|server|node|source|http|module|wordpress plugin|ruby|implementat
ion|html|repo|plugin mirror|personal|tool|scripts|version|page|file|demo|tools|projects|testing|rails|jquery|e
xtension|example|theme|wrapper|development|script


Cluster 1 :
|simple|using|library|application|python|line|command|framework|command line|tool|game|written|script|server|c
lient|plugin|javascript|example|node|android|wrapper|simple python|jquery|implementation|data|simple applicati
on|java|html|project|simple tool|file|simple library|simple script|simple game|http|simple wrapper|simple exam
ple|class|code|simple javascript|interface|ruby|django|files|module|demo|service|create|easy|text


Cluster 2 :
|blog|personal blog|personal|jekyll|octopress|github|source|octopress blog|site|website blog|website|tech|blog
site|code|jekyll blog|theme|repository|powered|static|github blog|static blog|portfolio|hexo|blog powered|pers
onal website|http|based|blog http|pages|page|ghost|source code|wordpress|code blog|pelican|django|blog engine|
just|http blog|development|official|test|engine|rails|repo|application|simple blog|developer|github pages|gene
rated


Cluster 3 :
|based|simple|game|framework|library|client|python|java|application|server|node|engine|tool|text|html|project|
based game|theme|text based|implementation|using|javascript|data|java based|source|http|python based|bootstrap
|code|framework based|browser|android|theme based|open|file|arduino|game based|template|management|library bas
ed|django|based application|plugin|browser based|open source|editor|model|user|jquery|software


Cluster 4 :
|project|final|final project|repository|test project|test|course|code|group|class|spring|sample|group project|
sample project|data|website|java|university|game|template|android|repo|software|school|project repository|cour
se project|design|demo|project using|using|demo project|team|development|school project|example|source|open|fi
les|django|class project|database|github|example project|server|django project|project template|application|ja
va project|android project|framework
```

After seeing the cluster contents we can name the clusters as following:

Cluster 0: Websites Cluster

This cluster contains information related to the technologies used for building clusters and different key words related to websites

Cluster 1: Frameworks and Tools Cluster

This cluster contains different types of frame works, technologies related to the framework and key words for frameworks.

Cluster 2: Code Cluster

This cluster contains different key words for code and code storage files such as libraries, files, repositories, source code, mirror etc.

Cluster 3: Project Cluster

This is a project cluster, the most important cluster we have been looking for. It has all the types of projects related to class, capstone, academic and group projects.

Cluster 4: Server Cluster

This cluster contains different types of server and server based scripts. It has a lot of keywords related to server such as client, remote,setup, socket etc.

In [152]:
```
# Getting the tfidf highest and lowest tfidf values and terms from each cluster
for i in range(0,4):
    print('\n')
    print("Cluster {} :".format(i))
    tfidf_dict={}
    for j in order_centroids[i,:10]:
        tfidf_dict[j]=tfidf[j]
    import operator
    sorted_tfidf = sorted(tfidf_dict.items(), key=operator.itemgetter(1))
    min_keys=[]
    first_word_key=sorted_tfidf[1][0]
    second_word_key=sorted_tfidf[2][0]
    max_keys=sorted_tfidf[1][-1]
    print("{},{}".format(tfidf[first_word_key],tfidf[second_word_key]))
    print("{},{}".format(terms[first_word_key],terms[second_word_key]))
#     #min_key=min(tfidf_dict, key=tfidf_dict.get)
    #print('The max and min tfidf value for the features in the cluster {} are: {} and {}'.format(i,max(tfidf_di

    #print(max_key)
    #print('High tfidf word: {}'.format(terms[max_key]))
    #print('Low tfidf word: {}'.format(terms[min_key]))
```

```
Cluster 0 :
4.313823681780586,4.346839657695938
using,library


Cluster 1 :
4.313823681780586,4.346839657695938
using,library


Cluster 2 :
5.117621655712726,5.263572158863268
github,site


Cluster 3 :
4.346839657695938,4.5184967019085995
library,based
```

In [153]:
```
tfidf_array=[]
for j in order_centroids[0,:]:
    tfidf_array.append(tfidf[j])

max(tfidf_array)
```

Out[153]:
```
11.753329576390485
```

In [154]:
```
# predicting from the fitted k means model
kmeans_clusters = kmeans_model.predict(vz)
print(len(kmeans_clusters))
```

```
514640
```

In [155]:
```
#adding the cluster labels to the dataframe
df_repo['Cluster_Code']=kmeans_clusters
df_repo.head()
```

```
/home/isiia/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
```

Out[155]:

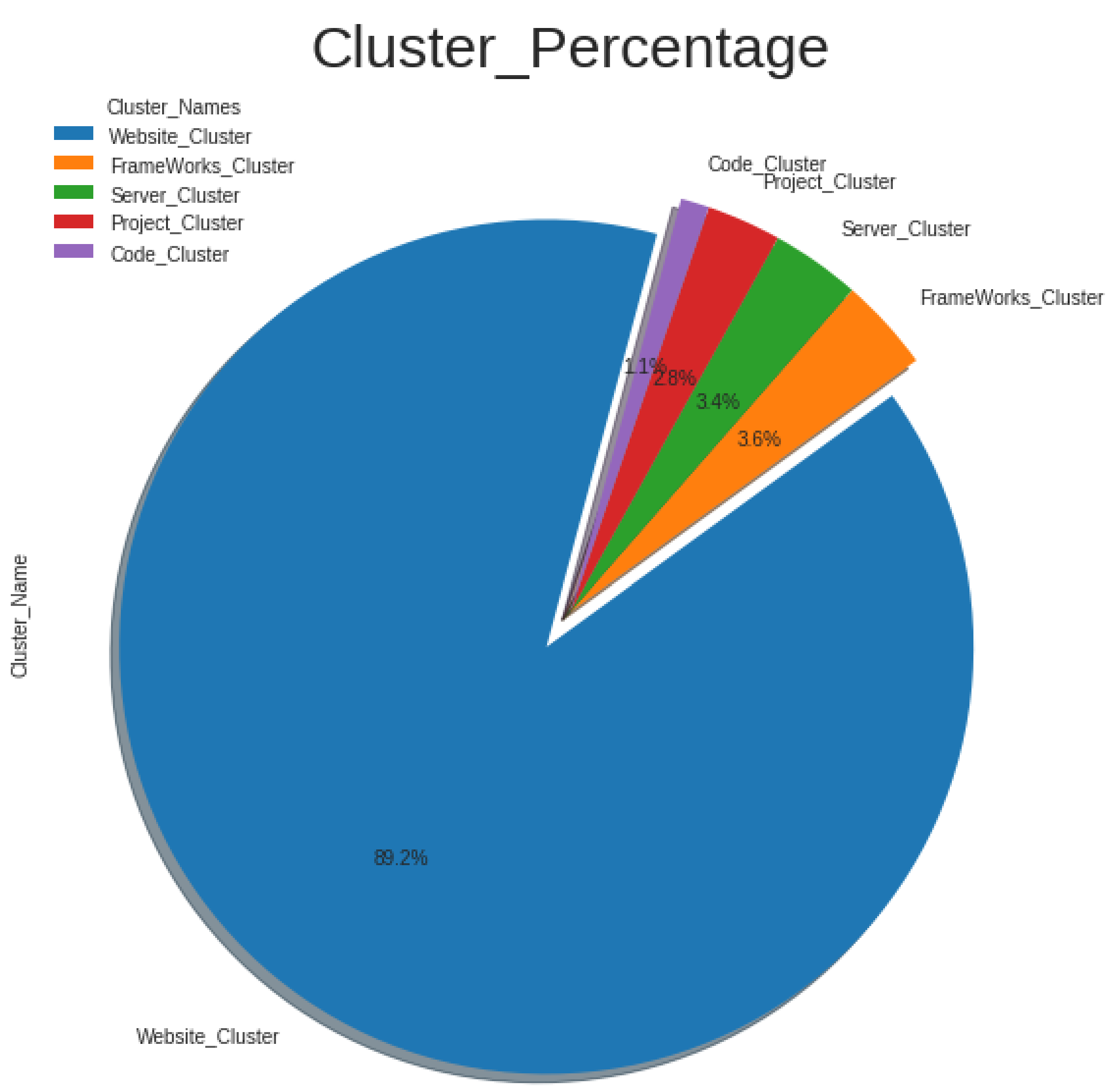
	repository_name	repository_description	repository_language	cleaned_text	tokens	Cluster_code	Cluster_Code	Cluster_Name
0	Intro-to-CSS3-Properties	CSS3 fun. Box shadows, opacity, border-radius,...	JavaScript	shadows opacity border radius transitions	[css, fun, box, shadows, opacity, border, radi...	0	0	Website_Cluster
1	A_new_dock	A OSX style dock using CSS and a few lines of ...	CSS	style dock using lines jquery	[osx, style, dock, using, css, lines, jquery]	0	0	Website_Cluster
2	rscanf	Read repeated input from a file or stdin.	C	read repeated input from file stdin	[read, repeated, input, file, stdin]	0	0	Website_Cluster
3	qstoq-item-frontend	qstoq item form app	CSS	qstoq item form	[qstoq, item, form, app]	0	0	Website_Cluster
4	Intro-to-Responsive-Design	Learn about media queries and responsive design	JavaScript	learn about media queries responsive design	[learn, media, queries, responsive, design]	0	0	Website_Cluster

In [156]:
```
Cluster_dict={0:'Website_Cluster',1:'FrameWorks_Cluster',2:'Code_Cluster',3:'Project_Cluster',4:'Server_Cluster'}
```

In [157]:
```
df_repo['Cluster_Name']=df_repo["Cluster_Code"].apply(lambda c: Cluster_dict[c])
```

```
/home/isiia/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  """Entry point for launching an IPython kernel.
```

In [158]:
```
#Exploring the clusters in our dataset

#df_repo.Cluster_Name.value_counts().plot.pie();


#plotting a pie chart to show datatypes
plt.style.use('seaborn-dark')
colors=['#8b3058','#fdfbe4','red','green']
startangle=75
explode=[0.1,0,0,0,0]
plt.rcParams['axes.titlesize']=30
plt.figure(figsize=(10,10));
df_repo.Cluster_Name.value_counts().plot.pie(explode=explode,title="Cluster_Percentage",autopct='%1.01f%%',starta
plt.legend(title="Cluster_Names");
plt.savefig('/home/isiia/Github_Classification/Data_Visualization/Cluster_pie')
```



In [ ]:
```
# silhouette_score(vz,kmeans_clusters)
```

In [161]:
```
x=df_repo.groupby('Cluster_Name').get_group('Server_Cluster').repository_description.tail(6)
```

In [162]:
```
x
```

Out[162]:
```
514526    The Biological Observation Matrix (BIOM) Forma...
514538               Team 16's Capstone Programming Project
514539                     Capstone Project App Academy
514542                                         Project4
514582                          sloodle project website
514615    shared repository for cassia project developments
Name: repository_description, dtype: object
```

In [159]:
```
#Lets see the cluster numbers beside the repository descriptions
kmeans_df=pd.DataFrame(kmeans_clusters)
kmeans_df.columns=['Cluster_labels']
kmeans_df=pd.concat([df_repo,kmeans_df],axis=1)
kmeans_df.head()
```

Out[159]:

	repository_name	repository_description	repository_language	cleaned_text	tokens	Cluster_code	Cluster_Code	Cluster_Name	Cluster_labels
0	Intro-to-CSS3-Properties	CSS3 fun. Box shadows, opacity, border-radius,...	JavaScript	shadows opacity border radius transitions	[css, fun, box, shadows, opacity, border, radi...	0	0	Website_Cluster	0
1	A_new_dock	A OSX style dock using CSS and a few lines of ...	CSS	style dock using lines jquery	[osx, style, dock, using, css, lines, jquery]	0	0	Website_Cluster	0
2	rscanf	Read repeated input from a file or stdin.	C	read repeated input from file stdin	[read, repeated, input, file, stdin]	0	0	Website_Cluster	0
3	qstoq-item-frontend	qstoq item form app	CSS	qstoq item form	[qstoq, item, form, app]	0	0	Website_Cluster	0
4	Intro-to-Responsive-Design	Learn about media queries and responsive design	JavaScript	learn about media queries responsive design	[learn, media, queries, responsive, design]	0	0	Website_Cluster	0

In [ ]:

Plotting cluster by selecting a sample

In [73]:
```
#For Sample

# I am selecting the number of clusters as 18 just a feeling because the number of target variables is 18 initia

# After using the elbow plot i selected the number of clusters value as 7
num_clusters = 7 # need to be selected wisely

# initializing the mini batch Kmeans model
kmeans_model_sample = MiniBatchKMeans(n_clusters=num_clusters,
                               init='k-means++',
                               n_init=1,
                               init_size=1000, batch_size=1000, verbose=0, max_iter=1000)
```

In [74]:
```
kmeans = kmeans_model_sample.fit(vz_sample)
kmeans_clusters_samples = kmeans_model_sample.predict(vz_sample)
kmeans_distances_samples = kmeans_model_sample.transform(vz_sample)
print(kmeans_distances.shape)
# reduce dimension to 2 using tsne
tsne_kmeans = tsne_model.fit_transform(kmeans_distances_samples)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-74-b9010ece8230> in <module>
----> 1 kmeans = kmeans_model_sample.fit(vz_sample)
      2 kmeans_clusters_samples = kmeans_model_sample.predict(vz_sample)
      3 kmeans_distances_samples = kmeans_model_sample.transform(vz_sample)
      4 print(kmeans_distances.shape)
      5 # reduce dimension to 2 using tsne

NameError: name 'vz_sample' is not defined
```

In [52]:
```
#setting colors
# spectral = np.hstack([Spectral6] * 20)
# colors = [spectral[i] for i in range(0,num_clusters)]
# print(colors)
#colors={0:'red',1:'green',2:'blue', 3:'black',4:'yellow',5: 'pink',6:'brown',7: '', 8: '#fee08b',9:'#fc8d59',10
```

In [53]:

```python
# select a palette
from bokeh.palettes import Dark2_5 as palette
# itertools handles the cycling
import itertools

# create a color iterator
colors={}
colors_pal = itertools.cycle(palette)
for m, color in zip(range(0,num_clusters), colors_pal):
    colors[m]=color

print(colors)
```

```
{0: '#1b9e77', 1: '#d95f02', 2: '#7570b3', 3: '#e7298a', 4: '#66a61e', 5: '#1b9e77', 6: '#d95f02'}
```

In [54]:

```python
# we made the 18 columns from the kmeans distance matrix to 2 columns (distance means the distance of that point

# making a dataframe with reduced kmeans distances
kmeans_df = pd.DataFrame(tsne_kmeans, columns=['x', 'y'])
kmeans_df['cluster'] = kmeans_clusters_samples
kmeans_df['description'] = df_repo['cleaned_text']
kmeans_df['repository_language'] = df_repo['repository_language']
kmeans_df['tokens']=df_repo['tokens']
#kmeans_df['cluster']=kmeans_df.cluster.astype(str).astype('category')
kmeans_df["color"] = kmeans_df["cluster"].apply(lambda c: colors[c])
```

In [55]:

```python
#plotting using bokeh
output_notebook()
source = ColumnDataSource(data=dict(x=kmeans_df['x'], y=kmeans_df['y'],
                                    description=kmeans_df['description'],
                                     repository_language=kmeans_df['repository_language'],cluster=kmeans_df['clu
```

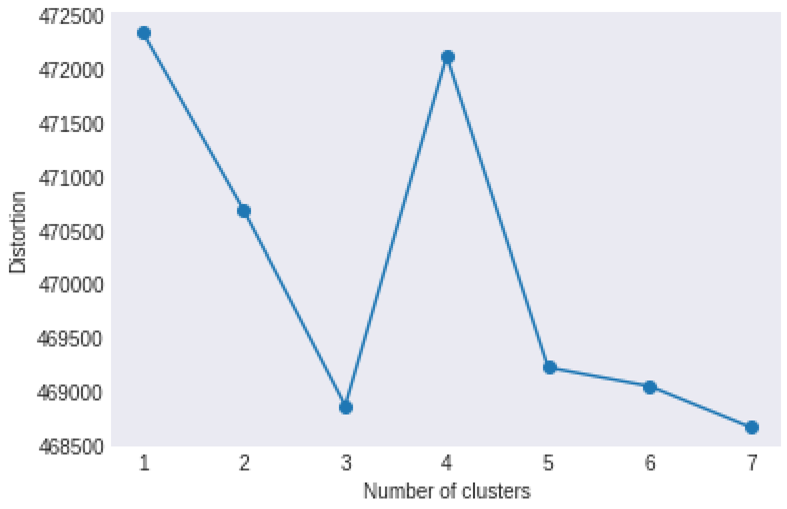
[(https://bokeh.pydata.org)](https://bokeh.pydata.org) Loading BokehJS ...

In [56]:

```python


plot_kmeans = figure(plot_width=700, plot_height=600,title="Clustering of the repository description",
    tools="pan,wheel_zoom,box_zoom,reset,hover,previewsave",
    x_axis_type=None, y_axis_type=None, min_border=1)

#colormap

spectral = np.hstack([Spectral6] * 20)
colors = [spectral[i] for i in range(1,num_clusters)]
plot_kmeans.scatter(x='x',y='y',fill_color='color',source=source)
#adding interactivity
hover = plot_kmeans.select(dict(type=HoverTool))
hover.tooltips={"description": "@description", "Cluster": "@cluster","tokens":"@tokens", "repo_language":"@repos
show(plot_kmeans)
```

The elbow method is a useful graphical tool to estimate the optimal number of clusters k for a given task. Intuitively, we can say that, if k increases, the within-cluster SSE ("distortion") will decrease. This is because the samples will be closer to the centroids they are assigned to.

In [91]:
```
# Selecting the optimal number for k (Elbow method)
# calculate distortion for a range of number of cluster
num_clusters=8
distortions = []
for i in range(1, num_clusters):
    km = MiniBatchKMeans(n_clusters=i,
                               init='k-means++',
                               n_init=1,
                               init_size=1000, batch_size=1000, verbose=0, max_iter=1000,random_state=8)
    km.fit(vz)
    distortions.append(km.inertia_)

# plot
plt.plot(range(1, num_clusters), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.show()
```



From the elbow plot we understood that the optimum number of clusters is 6

In [65]:
```
df_repo.drop(['tokens'],inplace=True,axis=1)
```

```
/home/isiia/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:3940: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  errors=errors)
```

In [66]:
```
# storing the cleaned text data into a text data base
df_repo.to_sql(con=engine, name='gitinfo_text', if_exists='replace',index=False,chunksize=1000)
```

In [61]:

In [ ]: