Technological Trends Evaluator and Predictor

Using StackOverFlow Data

Sakshi Goel
PESU
5th sem

Balusa Venkata Sai Harika
RVCE
5th sem

TABLE OF CONTENTS

PROBLEM STATEMENT

Technology trends evaluator and predictor which can guide students and professionals to improve their technological capabilities.

ABSTRACT

This project performs analysis on the Stack Overflow dataset to find out the trends in various technologies from 2009 to 2018 and predict the future technological trends using Linear Regression.

These are some of the questions that this notebook aims to answer for now:

1. What is the trend in the technologies from 2009 - 2018?
2. What is the trend in the various categories of each technology?
3. What will be the upcoming trends in the technologies?
4. What will be the upcoming trend in the various categories of each technology?

The data was loaded as Google's BigQuery Dataset on a Kaggle kernel, where it was analyzed to summarize its main characteristics. For the analysis, a broad look at the existing data was taken, using visual and quantitative methods to get a sense of the story the data was telling. Clues were searched for that suggested the next logical steps, questions or areas of research. The data was cleaned and then visualized by making use of various visualization techniques such as bar charts, stacked graphs, scatter plots and line graphs. This exploratory data analysis was used to determine the most talked about technology today.

Concepts of regression, particularly Linear Regression, were used to train a model on a certain fraction of the existing dataset to achieve an acceptable accuracy and hence, predict the technological trends in the upcoming future.

DATASET

Stack Overflow is a large and trusted online forum used by professional and enthusiastic programmers to learn, share their programming knowledge, and build their careers.

Stack Overflow was chosen to provide the data required to build the evaluator and predictor in question. All of Stack Overflow's data is available on Google's serverless, highly-scalable, and cost-effective cloud data warehouse called BigQuery.

The project was built on a Kaggle notebook to take advantage of its processing power, where the bq_helper package was used to access the required dataset from BigQuery, which simplifies common read-only tasks in BigQuery by dealing with object references and unpacking result objects into pandas dataframes on. The data can be found under the BigQuery project named bigquery-public-data, within the dataset called stackoverflow.

PACKAGES USED

1) matplotlib

This is a 2D plotting library that produces publication quality figures for purposes of visualization. This allows us visual access to huge amounts of data in easily digestible visuals through several plots such as line, bar, scatter and histogram. Matplotlib.pyplot has been extensively used in this kernel to plot graphs and charts.

2) NumPy

NumPy is a library that adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

3) bq_helper

The bq_helper package as described previously has been used to access the required Stack Overflow data.

4) pandas

pandas is a library that is used for data manipulation and analysis. It offers data structures and operations for the manipulation of numerical tables and time series. Stack Overflow data obtained has been stored into dataframes provided by the pandas library.

5) pandas_profiling

pandas_profiling is a package that is used to speed up the process of exploratory data analysis. It is useful when the data is not cleaned and still requires further individualized adjustments and pandas_profiling helps to guide where to start and what to focus on during such adjustments.

6) sklearn

Scikit-learn, also known as sklearn, features various algorithms for classification, regression and clustering. It is designed to interoperate with numerical library NumPy. This project uses sklearn to perform data splitting into train and test data and accordingly perform linear regression on it for purposes of training and prediction.

7) wordcloud

This visualization package is used to generate an image with a collection of words where the text size of each of the words is determined by their frequency or importance.

EXPLORATORY DATA ANALYSIS

1) Tables of '*stackoverflow*' Dataset -

The dataset has various tables under it, but this project particularly focuses on the table *'posts_questions'*.

```
stackoverflow.list_tables()
```

```
['badges',
 'comments',
 'post_history',
 'post_links',
 'posts_answers',
 'posts_moderator_nomination',
 'posts_orphaned_tag_wiki',
 'posts_privilege_wiki',
 'posts_questions',
 'posts_tag_wiki',
 'posts_tag_wiki_excerpt',
 'posts_wiki_placeholder',
 'stackoverflow_posts',
 'tags',
 'users',
 'votes']
```

2) '*posts_questions*' table schema -

	name	type	mode	description
0	id	INTEGER	NULLABLE	None
1	title	STRING	NULLABLE	None
2	body	STRING	NULLABLE	None
3	accepted_answer_id	INTEGER	NULLABLE	None
4	answer_count	INTEGER	NULLABLE	None
5	comment_count	INTEGER	NULLABLE	None
6	community_owned_date	TIMESTAMP	NULLABLE	None
7	creation_date	TIMESTAMP	NULLABLE	None
8	favorite_count	INTEGER	NULLABLE	None
9	last_activity_date	TIMESTAMP	NULLABLE	None
10	last_edit_date	TIMESTAMP	NULLABLE	None
11	last_editor_display_name	STRING	NULLABLE	None
12	last_editor_user_id	INTEGER	NULLABLE	None
13	owner_display_name	STRING	NULLABLE	None
14	owner_user_id	INTEGER	NULLABLE	None
15	parent_id	STRING	NULLABLE	None
16	post_type_id	INTEGER	NULLABLE	None
17	score	INTEGER	NULLABLE	None
18	tags	STRING	NULLABLE	None
19	view_count	INTEGER	NULLABLE	None

Every question posted on Stack Overflow is associated with a unique integer *'id'* that functions as the primary key for the *'posts_questions*' table. Every question posted has a title and a body as written by the user. The 'title' field is used to provide a short overview of the question while the '*body*' field includes a detailed question. Stack Overflow associates counts for the number of answers, comments, favorites and views for each question. These values are accessible through the fields '*answer_count', 'comment_count', 'favorite_count*' and '*view_count*' respectively. The date of posting of each question is stored in the '*creation_date*' field and this is a crucial attribute that has been considered for all the prediction purposes in this project. Based on the content of the question a user can associate his question with a number of tags stored in the '*tags'* field. This field can be used to selectively query posts for various purposes. Each post is also associated with several other attributes such as '*score', 'last_edit_date*' and '*owner_display_name'*.

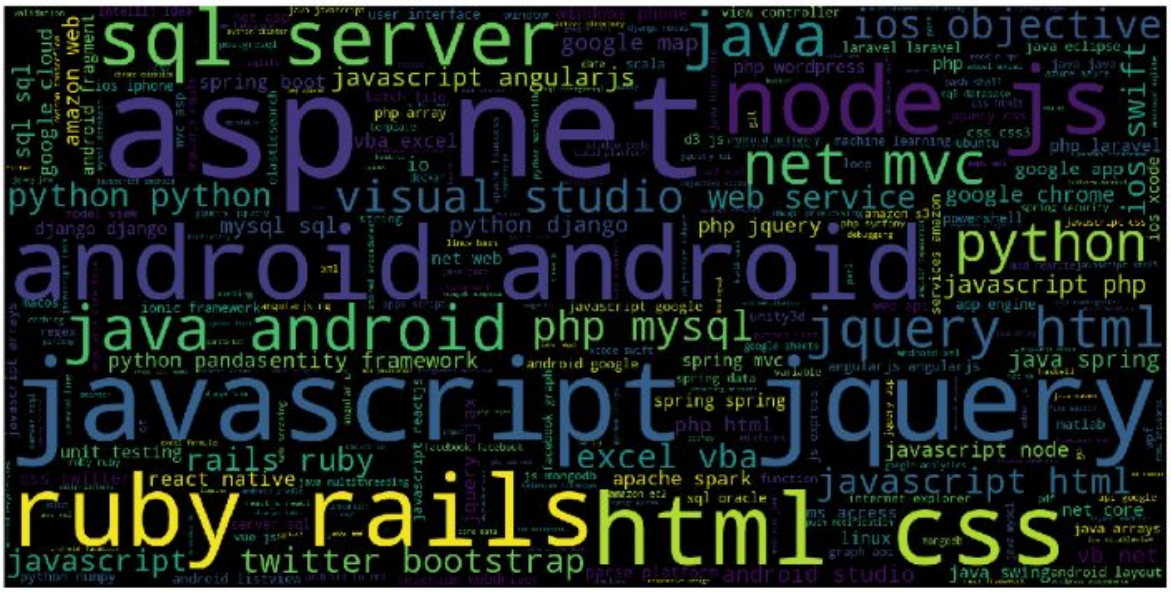
3) Sample rows of the table -

The top rows of a table can be viewed using the '*head*' function.

	id	title	body	accepted_answer_id	answer_count	comment_count	tags	view_count
0	9804136	How do I clear an array of a structure in C, a...	Two questions: \n\n \n How to quic...	9804289.0	3	2	c|struct	6462
1	44896935	How to implement threading to run two bash she...	I have to record a wav file and at the same...	44897223.0	2	6	python|multithreading|bash|shell|python-multit...	411
2	3059091	How to remove carriage returns from output of ...	I am using wordpress as a CMS and trying to...	3592816.0	7	5	php|javascript|wordpress|google-maps	73286
3	8926063	Code Coverage: Why is end marker red (End If, ...	I use MS-Test with Visual Studio 2010 and V...	8934375.0	4	0	vb.net|visual-studio|mstest|code-coverage	566
4	52977342	incorrect checksum for freed object - object w...	I was hoping the return values in c++11 are...	NaN	0	10	c++|c++11	203

4) WordCloud -

The following word cloud has been designed based on all the tags that have been used in Stack Overflow over the considered years and directly reflects the absolute popularity of any technology or software as the sum of the number of times a particular tag appeared in a question.



5) Data Profiling

A profile report of the uncleaned data may be generated using the pandas_profiling package. It gives an extensive analysis of all the attributes and rows in the particular pandas dataframe.

6) Data Cleaning

The irrelevant data is removed by querying the required, useful data from the dataframe and this can be achieved by using the '*query_to_pandas*' function as follows :

```
query = """select EXTRACT(year FROM creation_date) AS year, sum(id) as posts
        from `bigquery-public-data.stackoverflow.posts_questions`
        where extract(year from creation_date) >=2009 and extract(year from creation_d
ate) < 2019 and tags like '%javascript%'
        group by year
        order by year
        """

JavaScriptPosts = stackoverflow.query_to_pandas(query)
```

The missing values are taken care of by replacing them with an estimated value. The function *'fillna'* can be used to replace any null or missing values in any of the columns of the dataframe and the function *'df.isnull.sum()'* can be used to find the count of missing values, i.e. null values, if any, in all the columns of the dataframe.

TRENDS EVALUATION

Introduction

The years under consideration for the analysis of past trends are from 2009 to 2018. Broad technologies have been selected and under each of them, a couple of sub-categories have been chosen for analysis purposes :

1) Web Development

- AngularJs
- Bootstrap
- PHP
- HTML
- JavaScript
- CSS

2) Database Technologies

- MySQL
- NoSQL
- MongoDB
- PostgreSQL
- Cassandra

3) Big Data

- Hadoop
- Spark
- Hive
- HBase
- Kafka

4) Data Science

- Matplotlib
- Pandas
- Seaborn
- Regression
- SVM
- Kaggle

5) Programming Languages

- Ruby
- C++
- Python
- C#
- Java

The number of questions that have been posted related to a particular technology as a percentage of the total number of questions posted in that year has been selected as the parameter determining its popularity.

The year and the total number of posts per year posted on Stack Overflow have been queried as follows:

```
queryx = """select EXTRACT(year FROM creation_date) AS year, sum(id) as posts
        from `bigquery-public-data.stackoverflow.posts_questions`
        where extract(year from creation_date) >= 2009 and extract(year from creation_date) < 2
019
        group by year
        order by year
        """

PostsCount= stackoverflow.query_to_pandas(queryx)
```

The result of the query has been converted into a pandas dataframe *'PostsCount'*, which is as follows:

	year	posts
0	2009	423146198949
1	2010	2291741332650
2	2011	7963221293709
3	2012	18728621825482
4	2013	35927322690527
5	2014	52442814208513
6	2015	68888566178282
7	2016	84151286908611
8	2017	95369585214760
9	2018	102672988107002

For every chosen technology, the number of posts per year has been queried and then converted to a percentage of posts with respect to the total number of posts each year.

For example, for Web Development it has been done as follows :

```
query = """select EXTRACT(year FROM creation_date) AS year, sum(id) as posts
        from `bigquery-public-data.stackoverflow.posts_questions`
        where extract(year from creation_date) >=2009 and extract(year from creation_date) < 20
19 and (tags like '%bootstrap%' or
        tags like '%angularjs%' or tags like '%php%' or tags like '%html%' or tags like '%javas
cript%' or tags like '%css%')
        group by year
        order by year
        """

WebDevPosts = stackoverflow.query_to_pandas(query)
WebDevPosts['posts']= WebDevPosts['posts']*100/PostsCount.posts
```

The query has been written to find the sum of all the questions posted, that have tags of the subcategory technologies. These posts have been grouped by year and then ordered in ascending order based on the numerical value of the year. This result has been converted into the *'WebDevPosts'* dataframe. The '*Posts*' column of this dataframe holds the percentage value of the web development related posts taken over the total number of posts that year i.e *'PostsCount'*.

'WebDevPosts' gives the following result :

	year	posts
0	2009	14.799058
1	2010	17.036856
2	2011	19.276332
3	2012	20.557690
4	2013	22.839008
5	2014	24.517801
6	2015	24.681437
7	2016	24.199212
8	2017	22.746055
9	2018	19.759909

After this has been done, for each subcategory of a particular technology, the percentage of posts over the years has been calculated from the queried data of each of them and compared by constructing a multiple line graph with the *'Posts %'* on the y-axis and *'Year'* on the x-axis. This graph shows how the popularity of a particular technology increases or decreases over the years and can be used to analyze which new technologies have come into the picture and which ones have become old fashioned.

For example, for JavaScript under Web Development, the following query was run:

```
query = """select EXTRACT(year FROM creation_date) AS year, sum(id) as posts
        from `bigquery-public-data.stackoverflow.posts_questions`
        where extract(year from creation_date) >=2009 and extract(year from creation_date) < 20
19 and tags like '%javascript%'
        group by year
        order by year
        """

JavaScriptPosts = stackoverflow.query_to_pandas(query)
JavaScriptPosts['posts']= JavaScriptPosts['posts']*100/PostsCount.posts
pd.to_numeric(JavaScriptPosts['year'])
JavaScriptPosts
```

Here 'JavaScriptPosts' dataframe holds the percentage of posts for each year. The 'year' column of this dataframe has been converted to numeric type to help in carrying out predictions based on the past trends.

The resulting dataframe is as follows:

	year	posts
0	2009	5.596910
1	2010	6.395492
2	2011	7.706441
3	2012	8.448292
4	2013	9.725935
5	2014	11.014401
6	2015	11.662546
7	2016	11.992194
8	2017	11.789585
9	2018	10.857628

Past Trends Visualisations

Under each technology, the popularity of the subcategories has been compared by constructing a multiple line graph. Each of the subcategories' dataframes has been merged together into one dataframe to construct this graph. Each line corresponds to one of the subcategories and is differentiated by color which can be identified with the help of the legends table provided next to the graph.

This visualization of past trends has been performed for each of the chosen technologies as follows :

1) Web Development

The categories under consideration as mentioned previously are PHP, HTML, JavaScript, AngularJS, Bootstrap and CSS. Each of their corresponding dataframes has been merged together into one dataframe and the required multiple line graph for comparison of their popularity was constructed as follows:

```
WebDev= pd.merge(PHPPosts, htmlPosts, how='inner', on = 'year')
WebDev=WebDev.set_index('year')
WebDev= pd.merge(WebDev, JavaScriptPosts, how='inner', on = 'year')
WebDev =WebDev.set_index('year')
WebDev=pd.merge(WebDev,AngularJSPosts,how='inner',on='year')
WebDev = WebDev.set_index('year')
WebDev=pd.merge(WebDev,BootstrapPosts,how='inner',on='year')
WebDev = WebDev.set_index('year')
WebDev=pd.merge(WebDev,CSSPosts,how='inner',on='year')
WebDev = WebDev.set_index('year')

WebDev.plot(kind='line')
plt.xlabel('Year', fontsize=15)
plt.ylabel('Posts %', fontsize=15)
y_pos=[2009,2010,2011,2012,2013,2014,2015,2016,2017,2018]

plt.xticks(y_pos,fontsize=10)
plt.yticks(fontsize=10)
plt.title('Web Development')
plt.legend(['PHP','HTML','JavaScript','AngularJS','BootStrap','CSS'],loc=[1.0,0.5])
plt.show()
```

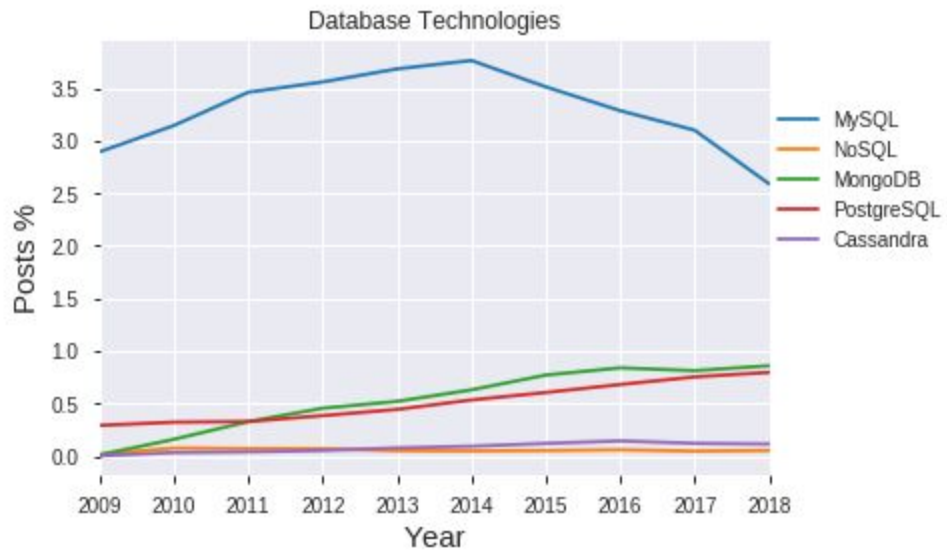
This gives us the following line graph :

From this, it can be seen that the popularity of JavaScript has greatly increased over the years. During the beginning of the decade, PHP was more popular as compared to the other technologies but as the years proceed the popularity decreased. The least popular technology is BootStrap whose popularity remains almost constant over the decade, as observed.

2) Database Technologies

The categories discussed here are MySQL, NoSQL, MongoDB, PostgreSQL and Cassandra. Each of their corresponding dataframes has been merged as described previously to construct the required multiple line graph, which is helpful in understanding the trends of these categories over the decade.

The following graph is obtained:

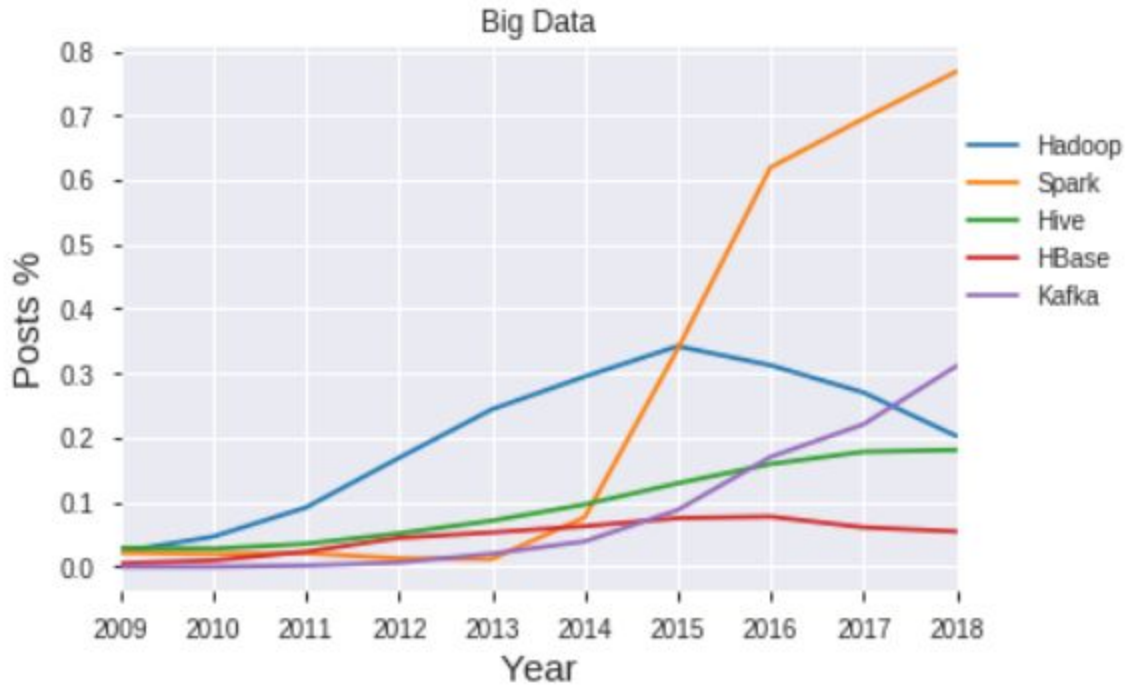


It is observed that even though its popularity is currently decreasing, MySQL is the favored database technology throughout the decade as compared to the others. As Cassandra is one type of NoSQL technology, that might be the reason why the percentage of posts for NoSQL and Cassandra has been the same over the years. As it can be seen from the graph, the popularity of MongoDB and PostgreSQL is increasing as time progresses and both of these technologies appear to maintain the same trend. Overall, it shows as MySQL is the most popular and the reason for this might be the ease of use and convenience that MySQL provides. Similarly, these trends can be used to find such reasons which make one particular technology preferred over another.

3) Big Data

The technologies discussed under Big Data are Hadoop, Spark, Hive, HBase and Kafka. A line graph is constructed in a similar way as mentioned for the above technologies to compare the popularities of these various subcategories and hence, get some useful insights about the trends observed over the years.

The following graph is attained:

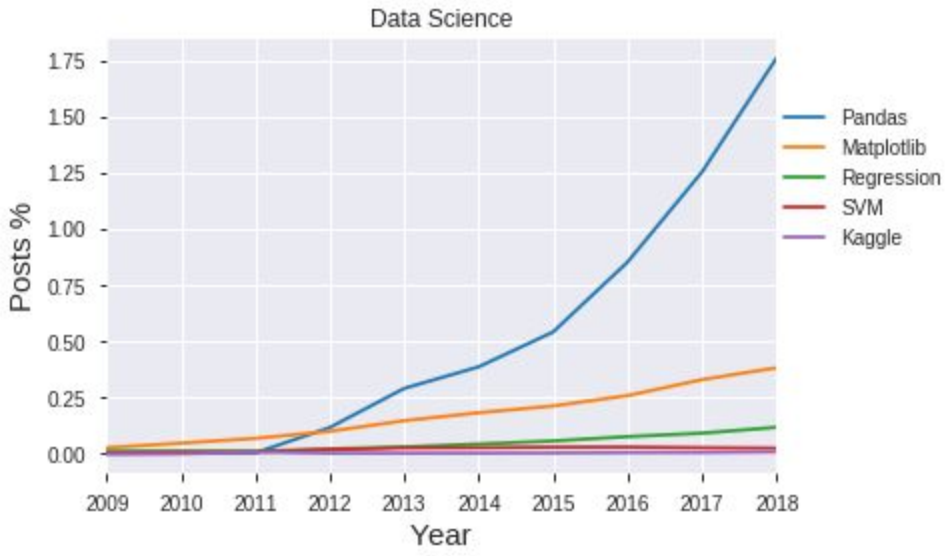


From the graph, one can infer that Hadoop was the most popular technology until 2015 but got overtaken by Spark. The popularity of Spark sharply increased around 2014 and it remains the most popular Big Data technology ever since. The popularity of HBase remains stagnant with no major peaks or pits. With the rise in demand for other technologies, Hadoop's popularity seems to stop low. Overall, all of these technologies under Big Data have an increasing slope and it can be concluded that Big Data is an upcoming and rising technology.

4) Data Science

Hadoop, Spark, Hive, HBase and Kafka are the technologies mentioned in this project under Data Science. Each of their corresponding dataframes has been merged as described previously to construct the required multiple line graph for comparison of their popularity.

This gives us the following graph:

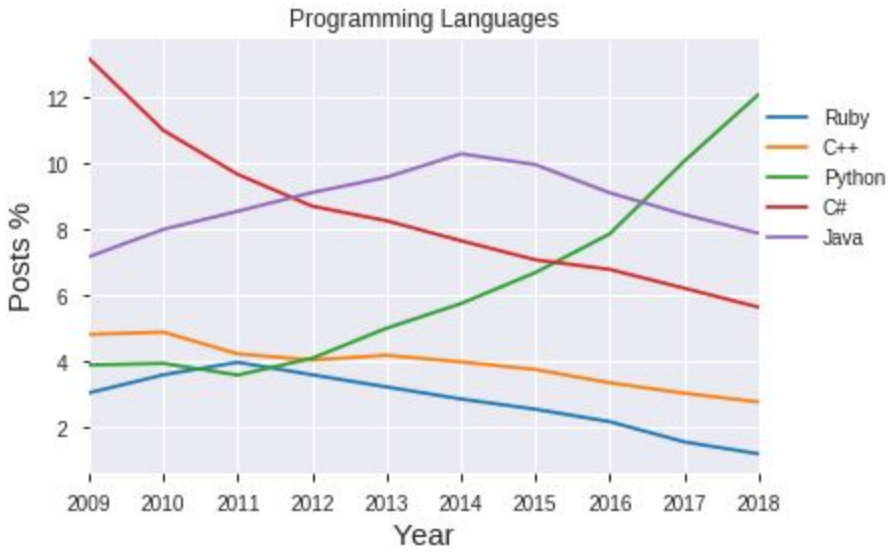


From the above graph, one observes that Regression, Kaggle and SVM are not as popular when compared to Pandas and Matplotlib. It is also seen that Kaggle started emerging around 2011. This might be caused due to the fact that Kaggle was founded in 2010. It is visible that the growth of Matplotlib's popularity is gradual while Pandas' increases exponentially 2011 onwards and it can be inferred that it is extensively used by the data science community.

5) Programming Languages

The programming languages considered in this project as mentioned previously are Ruby, C++, Python, C# and Java. The past data for these languages is queried and put into separate dataframes, which are then merged into one Programming Languages dataframe. This common dataframe is then used to construct the required multiple line graph for comparison of their popularity, in a similar way as the previous broad categories.

The resulting line graph is as follows :



It is observed that the popularity of languages C#, C++ and Ruby has fallen over the years while that of Python has seen a huge rise and is considered to be the most trending language as of now. Java became the most trending language in 2012 but its popularity started falling in 2014 and finally, Python took over around 2016-2017.

Past Trends Comparison

Each of the technologies can be compared to analyze past trends in a similar way through a multiple line graph. This has been done as follows:

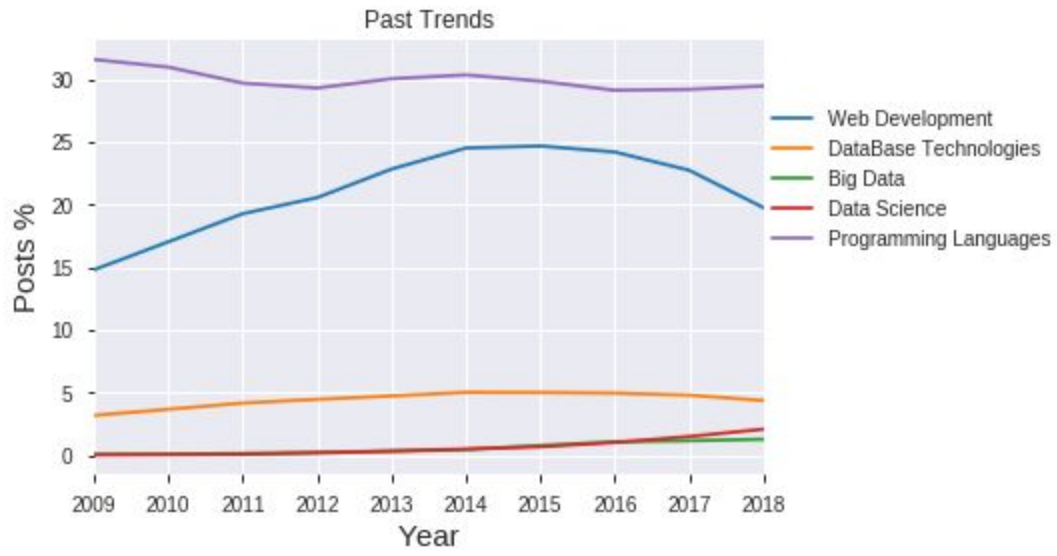
```
PastTrends= pd.merge(WebDev_Posts, DataBase_Posts, how='inner', on = 'year')
PastTrends =PastTrends.set_index('year')
PastTrends= pd.merge(PastTrends, BigData_Posts, how='inner', on = 'year')
PastTrends =PastTrends.set_index('year')
PastTrends=pd.merge(PastTrends,DataScience_Posts,how='inner',on='year')
PastTrends = PastTrends.set_index('year')
PastTrends=pd.merge(PastTrends,ProgLang_Posts,how='inner',on='year')
PastTrends = PastTrends.set_index('year')
```

The various dataframes of all the categories were merged into one dataframe, namely '*PastTrends*' and the index of this dataframe was set to the '*year*' column. This resulted in a dataframe where each of the various categories was one of the columns in this new dataframe.

This dataframe was plotted into a line graph using the Pandas.DataFrame.plot() function and the various attributes of this graph were changed using the functionality provided by the matplotlib.pyplot package.

```
PastTrends.plot(kind='line')
plt.xlabel('Year', fontsize=15)
plt.ylabel('Posts %', fontsize=15)
y_pos=[2009,2010,2011,2012,2013,2014,2015,2016,2017,2018]

plt.xticks(y_pos,fontsize=10)
plt.yticks(fontsize=10)
plt.title('Past Trends')
plt.legend(['Web Development','DataBase Technologies','Big Data','Data Science','Programming La
nguages'],
           loc=[1.0,0.5])
plt.show()
```

The graph obtained is as follows:



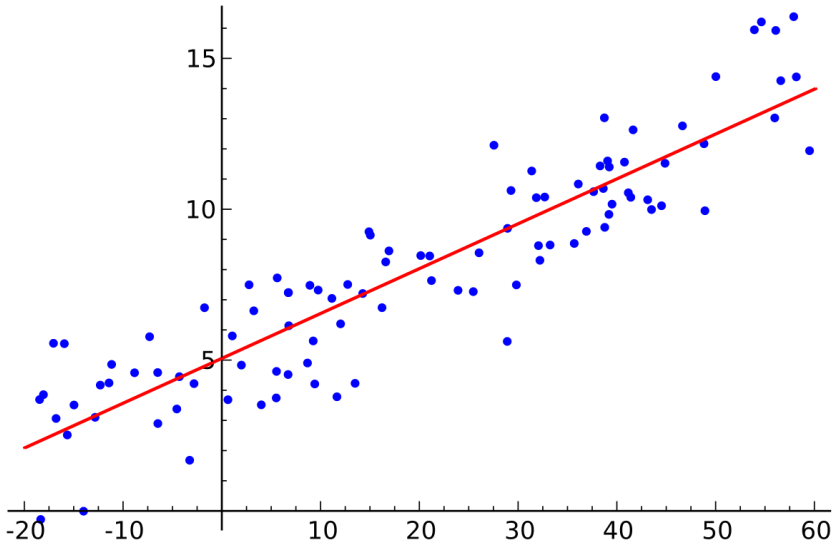
As it can be observed from the graph, Programming Languages are the most talked-about category as compared to the other categories, by a huge margin. It is followed by Web Development which reached its peak popularity around 2014-2015 after which its popularity started dipping. These are followed by Database Technologies, Data Science and Big Data respectively. Data Science and Big Data go hand in hand as they are practically implemented as one. This graph gives us a fair idea about which have been the most important technologies and are the most required in today's industry.

TRENDS PREDICTION

The trends are predicted for the year 2019. Several models exist that can be used for predictive analysis, but the Linear Regression model was chosen for this project due to its simplicity and usefulness.

Linear Regression

Linear Regression is a commonly used method for predictions and it is generally used with numerical data. It tries to model the relationship between the 2 variables by fitting a linear equation to the observed data. One variable is the independent or predictor variable while the other is the dependent or outcome variable. This model looks for a statistical relationship between the two variables. The main idea behind this model is to find a line that best fits the data. The best fit line is defined as the line for which the total prediction error is as small as possible. This error is calculated as the distance from a data point to the regression line.



Test and Train Set

The data available is split into training data and testing data using the '*train_test_split*' function. The prediction model built is trained on the training data and then used to predict values for the testing data. This is done so as to check the fitness of the model and find out its accuracy. One of the methods that have been employed for this purpose in this project is the mean squared error or the root mean squared error.

Mean Squared Error and Root Mean Squared Error

The mean squared error measures the average squared difference between the estimated values and the true values.

```
metrics.mean_squared_error(y_test, predictions)
```

```
9.611303240361987e+24
```

The root mean squared error is commonly used in regression analysis to verify experimental results by indicating how concentrated the data is around the best fit line.

```
np.sqrt(metrics.mean_squared_error(y_test, predictions))
```

```
3100210192932.4062
```

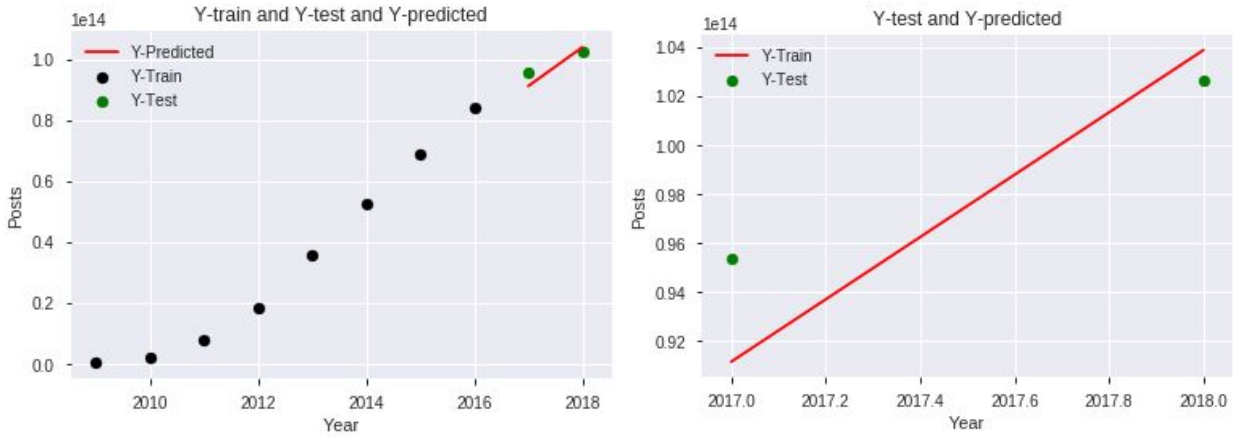
A model is accurate for a set of training data if the root mean squared error of the model for the testing data is less than 10% of the mean value. The mean of the columns can be found using the pandas.DataFrame.describe() function which provides all the statistics of the various columns of the Dataframe.

	year	posts
count	10.00000	1.000000e+01
mean	2013.50000	4.688593e+13
std	3.02765	3.935860e+13
min	2009.00000	4.231462e+11
25%	2011.25000	1.065457e+13
50%	2013.50000	4.418507e+13
75%	2015.75000	8.033561e+13
max	2018.00000	1.026730e+14

As observed, the mean of the '*posts*' column is 46.88 Tera units and the root mean squared error for our model was 3.1 Tera units, which is less than 10% of the mean. Hence, it can be concluded that the Linear Regression Model fits the data accurately.

Future Trends Visualisations

The training values, testing values and the predicted values for the '*PostsCount*' dataframe were plotted for a better understanding of the accuracy of the model on this data. As observed, the model predicted values very close to the expected values and hence, is a good model for this data.



The same visualizations were created for each of the major categories and it was observed that this model was not very accurate for prediction for the data of those categories as the trends for the individual categories did not follow a linear curve.

Future Trends Comparison

The following function is used to create bar graphs visualizations for the various technologies to compare the future trends amongst them. The datasets for the technologies or their subcategories can be passed as a list to this function and these datasets can be variable in number starting from 2. The labels for the various technologies and the title for the visualization can also be passed, but these fields are optional and are set to 'None' and 'Trends in Technologies in 2019' by default, respectively. 'Year' can also be passed as a parameter to the function and the function predicts the trends for the year passed or 2019 as default.

The function traverses through each of the dataframes passed in the list and trains a linear regression model on the existing data completely and then predicts the value for the passed year or 2019, by default. These predicted values of all the dataframes are stored in a predict list, which is then used to plot the bar graph.

The complete function code is as follows:

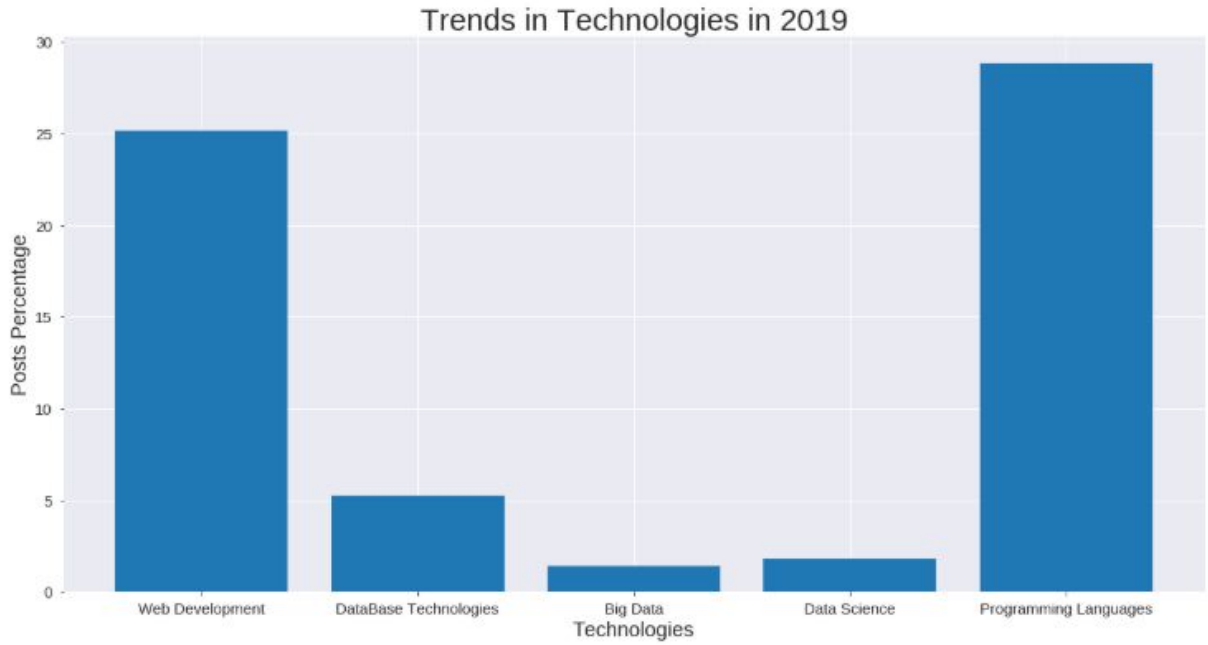
```
def trends(dfall, labels=None, Year = 2019, title="Trends in Technologies in ", **kwargs):

    plt.figure(figsize=(20,10))

    predict = []
    for df in dfall :
        year=df['year'].values.reshape(-1,1)
        posts=df['posts'].values.reshape(-1,1)
        reg=LinearRegression()
        X_train = year
        Y_train = posts
        X_test = [[Year]]
        reg.fit(X_train,Y_train)
        predictions = reg.predict(X_test)
        predict.append(predictions)

    trend = pd.DataFrame(columns = ['Technology','Posts %'])
    trend['Technology'] = labels
    trend['Posts %'] = predict

    x_pos = np.arange(len(trend['Technology']))
    plt.bar(x_pos,trend['Posts %'])
    plt.xticks(x_pos, trend['Technology'],fontsize=15)
    plt.yticks(fontsize=15)
    plt.xlabel('Technologies',fontsize=20)
    plt.ylabel('Posts Percentage',fontsize=20)
    plt.title(title+str(Year),fontsize=30)
    plt.show()
```

This function can be called as follows to get the future predictions for the year 2019 for all the 5 major technological categories discussed in this project:

```
trends([WebDev_Posts, DataBase_Posts, BigData_Posts, DataScience_Posts, ProgLang_Posts],
       ["Web Development",'DataBase Technologies','Big Data','Data Science','Programming Langua
ges'])
```

It can be noticed that a list with all the labels is also passed to the function, which takes care of all the x-labels in the graph. It is important to maintain the order of names in the labels list in accordance with the order in which the dataframes are passed in the dataframes list.

The previously called function results in the following bar chart:



From this, one can infer that the trendiest category in 2019 is predicted to be Programming Languages. This is a reasonable prediction as essentially every application in the field of Computer Science involves the use of some or the other programming language such as Java or Python. Following at a close second is Web Development which includes technologies like JavaScript and CSS. Nearly every deliverable that is made requires a web app or a desktop app which is developed using these technologies and hence is predicted to be widely used. It is observed that the popularity of DataBase Technologies is estimated to increase in 2019. The projected posts percentage of Big Data and Data Science for 2019 are seen to be nearly the same and this can be explained by the fact that the two technologies are interrelated and work together to achieve similar objectives.

The same function can be passed any of the technology dataframes as a list parameter and it will return a similar bar chart showing the expected percentage of the total posts that each of the technologies might have in the user-defined year or 2019, if not specified by the user.

GENERALIZED EVALUATOR

This evaluator function takes in a list of tags for which the user wants to find the past trends. These tags are queried within the function to get the past data for each tag and the queried dataframes are merged into one dataframe with each column as one tag. This dataframe is then used to plot a line graph that shows the past trends in the technologies mentioned as the tags list.

The function can also take optional parameters, namely labels and title. 'title' is set to 'Past Trends' by default, while the labels list is equal to the list of tags if not mentioned explicitly.

```
def PastTrends(dfall, labels = None, title="Past Trends", **kwargs):

    query1 = "select EXTRACT(year FROM creation_date) AS year, sum(id) as posts from `bigquery-
public-data.stackoverflow.posts_questions` where extract(year from creation_date) >=2009 and ex
tract(year from creation_date) < 2019 and tags like '%"
    query3 ="%' group by year order by year"
    df = []

    if labels==None:
        labels = dfall

    l = len(dfall)
    for i in range(l):
        query2 = dfall[i]
        query = query1+query2+query3
        Posts = stackoverflow.query_to_pandas(query)
        Posts['posts']= Posts['posts']*100/PostsCount.posts
        pd.to_numeric(Posts['year'])
        df.append(Posts)

    trend = pd.merge(df[0], df[1], how='inner', on = 'year')
    trend = trend.set_index('year')
    if(l>2):
        for i in range(2,l):
            trend = pd.merge(trend, df[i], how='inner', on = 'year')
            trend = trend.set_index('year')
```

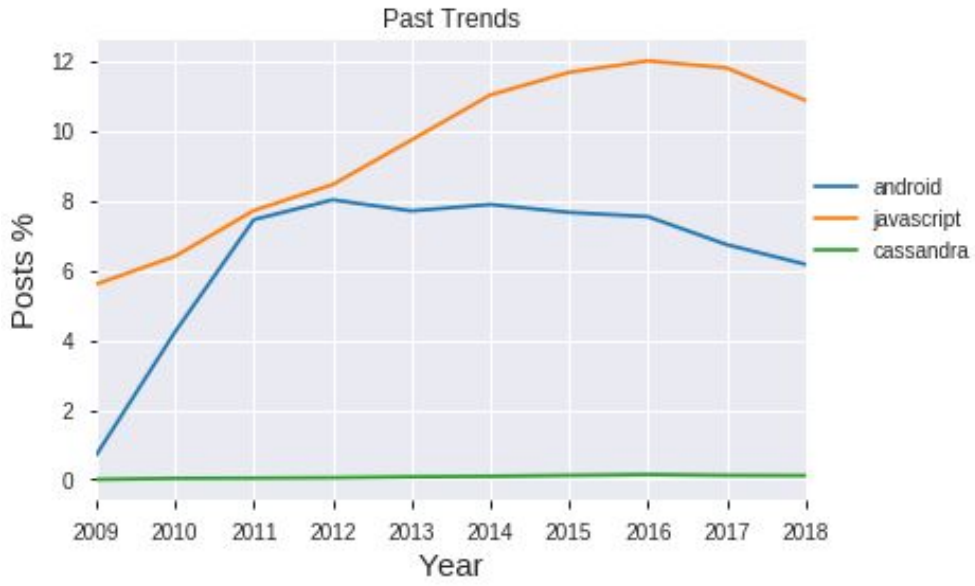
```
trend.plot(kind='line')
plt.xlabel('Year', fontsize=15)
plt.ylabel('Posts %', fontsize=15)
y_pos=[2009,2010,2011,2012,2013,2014,2015,2016,2017,2018]
plt.xticks(y_pos,fontsize=10)
plt.yticks(fontsize=10)
plt.title(title)
plt.legend(labels, loc=[1.0,0.5])
plt.show()
```

The function is called as follows:

```
PastTrends(["android","javascript","cassandra"])
```

This finds the past trends for the tags ‘android’, ‘javascript’ and ‘cassandra’. The legends are the same as the tags as a list with the labels has not been passed to the function call explicitly.

The following multiple line graph is obtained for the tags entered by the user :



From this graph, the user can clearly understand how the trends of the chosen technologies have varied over the period from 2009 to 2018.

In the same way, any number of tags can be passed as a list to the function and the function returns a multiple line graph depicting how the trends have varied over the decade for the technologies represented by the tags.

GENERALIZED PREDICTOR

This function takes in a list of tags for which the user wants to find the future trends. These tags are queried within the function to get the past data for each tag and a linear regression model is built for each of the tags and future predictions are made using the queried past data. A bar graph is created which shows the predicted future trends in the passed year for the given tags.

The function can also take optional parameters, namely labels and title. 'title' is set to 'Trends in Technologies in 2019' by default, while the labels list is equal to the list of tags if not mentioned explicitly. The function can also take 'year' as one of the arguments, which controls the year for which predictions are made. By default, 'year' is set to 2019.

```
def FutureTrends(dfall, Year = 2019, labels = None, title="Trends in Technologies in
", **kwargs):

    plt.figure(figsize=(20,10))

    query1 = "select EXTRACT(year FROM creation_date) AS year, sum(id) as posts from `
bigquery-public-data.stackoverflow.posts_questions` where extract(year from creation_d
ate) >=2009 and extract(year from creation_date) < 2019 and tags like '%"
    query3 ="%' group by year order by year"
    df = []
    l = len(dfall)

    if (labels==None):
        labels = dfall

    for i in range(l):
        query2 = dfall[i]
        query = query1+query2+query3
        Posts = stackoverflow.query_to_pandas(query)
        Posts['posts']= Posts['posts']*100/PostsCount.posts
        pd.to_numeric(Posts['year'])
        df.append(Posts)
```

The model built is a Linear Regression model, which is trained on the past trends data of the tags for the period 2009 to 2018, as mentioned earlier. The number of posts returned by the query is converted to the percentage of posts with respect to the total number of posts as stored in the '*PostsCount*' dataframe. All these dataframes for the tags are collected in a list, that is later used for prediction.

```
    predict = []
    for d in df:
        year=d['year'].values.reshape(-1,1)
        posts=d['posts'].values.reshape(-1,1)
        reg=LinearRegression()
        X_train = year
        Y_train = posts
        X_test = [[Year]]
        reg.fit(X_train,Y_train)
        predictions = reg.predict(X_test)
        predict.append(predictions)
    #print(predict)

    trend = pd.DataFrame(columns = ['Technology','Posts %'])
    trend['Technology'] = labels
    trend['Posts %'] = predict

    x_pos = np.arange(len(trend['Technology']))
    plt.bar(x_pos,trend['Posts %'])
    plt.xticks(x_pos, trend['Technology'],fontsize=15)
    plt.yticks(fontsize=15)
    plt.xlabel('Technologies',fontsize=20)
    plt.ylabel('Posts Percentage',fontsize=20)
    plt.title(title+str(Year),fontsize=30)
    plt.show()
```

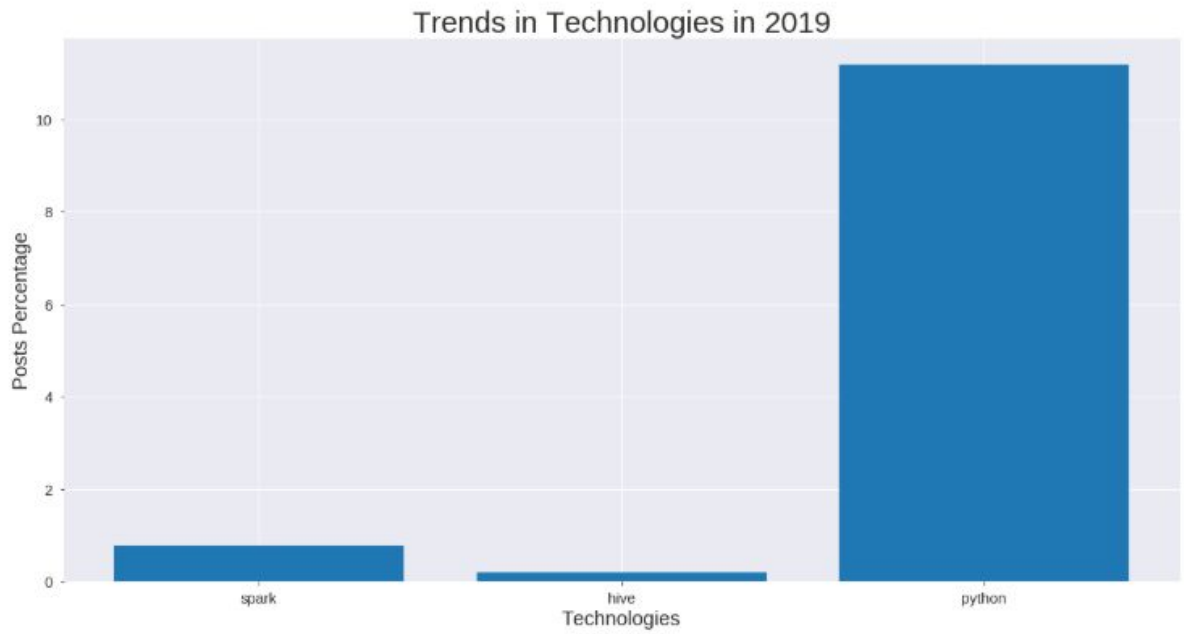
Once the training is done, the model predicts the percentage of posts for each of the tags. This value is appended onto a '*predict*' list. This list is used to plot the y-axis of the 'Future Trends' bar graph.

One simple example of the function's call is as follows:

```
FutureTrends(["spark","hive","python"])
```

This returns a bar graph showing the percentage of posts for the tags 'spark', 'hive' and 'python' in the coming year 2019. Since the list of labels has not been explicitly mentioned, the tag list is used as the y-labels for the graph. If '*Year*' was passed explicitly, the function would have returned a bar graph depicting future trends for that particular year, instead of 2019. The title of the graph will also be the default value unless passed explicitly to the function.

The following bar chart is obtained for the tags entered by the user:



It can be observed from this graph that python is expected to be the most recognized technology as compared to spark and hive, which have much lower popularities with hive having the lowest of them all.

From such graphs, the user can understand how popular the technologies passed into the function are going to be in the upcoming years.

FUTURE WORK

Currently, this project considers a single attribute 'years' to predict future trends. In the future, a better-suited model can be chosen by considering more relevant attributes to increase the accuracy of the predictions.

More intensive analysis can also be performed along with more customizations based on geographical regions or monthly trends. Additionally, a deliverable dashboard can be created as a web application that is more interactive.

CONCLUSION

By utilizing Stack Overflow's historical data made available through Google's BigQuery, this project successfully performs analysis to uncover past trends as well as predict future trends in technology by modeling an appropriate Linear Regression model. The resulting plots are self-explanatory and can be understood by anyone. This project serves to be useful for a wide range of users such as students, professors, those in search of a job or simply those that wish to be up to date with the current technology trends. It makes it easier for users to understand the trends simply by just calling the evaluator and predictor functions, instead of having to look into robust databases that do not provide readability. These functions can be tweaked to provide more functionality and implement other models. This provides a user-level abstraction and offers reusability.

RESOURCES

Project Code

[1] Kaggle: StackOverflow Technological Trends Evaluator and Predictor

[2] Github: StackOverflow_Technological_Trends

Reference Kernels

[1] Introduction to the bq_helper package | Kaggle

[2] BigQuery to analyse the Stack Overflow data | Kaggle

[3] State of Javascript in 2018 on Stackoverflow | Kaggle

[4] Python trends in 2018 | Kaggle

Reference WebPages

[1] Hatched symbols in matplotlib

[2] matplotlib.patches.Patch Python Example

[3] Customizing Plot Legends | Python Data Science Handbook

[4] 2019 Top Technology Trends | Based on Stackoverflow Data Analysis

[5] Exploratory Data Analysis (EDA) and Data Visualization with Python - Kite

[6] BARPLOT – The Python Graph Gallery