INTRODUCTION

1.1 INTRODUCTION

Software cost estimation is a critical stage that is being done in the initial phases of software development process. The main objective is to have clear project details and specifications to assist stakeholders in managing the project in terms of human resources, assets, software, data and even in the feasibility study. Accurate estimation results will definitely helps the project manager to do better estimation for the project cost. However, the inaccuracy may result from the project cost estimation process that will certainly affect the project delivery. A project with wrong or imprecision evaluation will face issues with delivery timing, resources required, budget or even in quality or operational side and sometimes the project may fail or aborted. Hence, the cost estimation is a significant part of the software projects and so it continues to be a complex issue in the software engineering field. Therefore, many studies and researches have been conducted for the purpose of enhancing and improving the estimation process and get more accurate and dependable results. Machine learning (ML) techniques become very essential in software studies. In many scientific researches, ML methods are being used and executed most likely in the various fields. ML can be a suitable technique to build the proposed model due to the ability to learn from historical data and adapt the wide variations that join software project development. In this work, ML techniques will be used to evaluate and compare the results of implementing such techniques on dataset. By applying ML methods on the dataset, it can be concluded if the ML techniques could be applied successfully on software cost estimation data or not. If yes, it would be possible to know which method scored the best results and also it is likely to decide if an ML model can be developed to evaluate and estimate the software cost.The aim of the project is to tackle these limitations and narrow the gap between up to date research findings and potential deployment of robust machine learning algorithms in practice for effort and duration estimation at the initial project lifecycle of software initiatives. Therefore, a comprehensive approach is presented, beginning from data preparation to the models' implementation and maintenance that ensures their usability as well as outstanding estimation accuracy and robustness for noise within data. For that purpose, a practical and effective approach for preparing data and building models is applied and presented based on the ISBSG dataset, which provides the most reliable source of a large volume of recent software projects from multiple industries (International Software Benchmarking Standards Group, 2013) with machine learning predictive algorithms.

1.2 OBJECTIVE

Software estimation is one of the most challenging areas of project management. The purpose of this project is to narrow the gap between up-to-date research results and implementations within organisations by proposing effective and practical machine learning deployment and maintenance approaches by utilization of research findings and industry best practices. This was achieved by applying ISBSG dataset, smart data preparation, machine learning algorithms (Support Vector Machines, Logistic Regression) and cross validation. For calculation of effort and duration SLOC are considered from the ISBSG dataset.

1.3 PROPOSED SYSTEM

The proposed system deals with estimation of effort and duration of various machine learning models on ISBSG dataset.The aim of this project is to compare machine learning method such as SVM, Logistic regression in terms of accuracy. This proposed system helps the industries to know the opinions of customers on their project and improve the quality of their projects. In this Project, Performance metrics such as F1,Recall,Precision and Support are calculated.Along with performance metrics, Regression metrics are also calculated .some of the regression metrics are MeanStandardError, MeanAbsoluteError, RootMeanSquareError are calculated. Based on SLOC (lines of code) cocomo estimation methods such as Organic, Semi-detached, embedded are calculated for effort and duration of software project estimation.There are many techniques that are present for handling noisy data.Noisy and unreliable data may severely influence the predictive accuracy of machine learning models. Poor quality of data. especially the significant occurrence of missing values and outliners may lead to inconsistent and unreliable results. Therefore, data preparation is a critical task in the process of building ML models in which data is preprocessed through selection, cleaning, reduction, transformation and feature. For SVM, It is found to be used Radial basis kernel. Support Vector Machine – kernel function: radial basis (RBF).

LITERATURE SURVEY

[1].Przemyslaw Pospieszny , ” ***An effective approach for software project effort and duration estimation with machine learning algorithms*** ”.

Considering the current dynamic software development environment with agility, prototyping and rapid delivery, software simulation using machine learning algorithms could further enhance project estimation methods and contribute to better resource allocation and utilization. In this paper ensembling with ML algorithms are implemented.

RBF kernel, mostly used in SVM classification, maps input space in indefinite dimensional space. Following formula explains it mathematically −

(i) K(x,xi)=exp(−gamma∗sum(x−xi^2))

$$\text{(ii)}\quad g(z) = \frac{1}{1+e^{-z}}$$

The Above formula is used for calculating the logistic regression and support vector machine with RBF kernal.

[2]. Jianglin Huang etal,” ***An empirical analysis of data preprocessing for machine learning-based software cost estimation***”.

Missing data is a common situation in software engineering datasets.Data preprocessing is a fundamental stage of ML method and has large impact on the accuracy of ML methods.An empirical study is conducted to analyze the effectiveness of the four DP techniques such as MDT, Scaling, FS and CS. The interactions between the preprocessing techniques and ML methods’ predictive accuracies are also studied in this paper.

$$[0,1]interval = \frac{actual\ value - \min(all\ values)}{\max(all\ values) - \min\ (all\ values)}$$

The above formula is used to calculate the scaling value.It is one of the way to handle missing values.

[3]. Rekha Tripathi et al, ''***Machine Learning Methods of Effort Estimation and It's Performance Evaluation Criteria***''.

A overview of machine learning estimation techniques and their strengths and weakness and also described performance evaluation criteria of estimation techniques. Machine learning approaches could be appropriate estimation technique because they can increase the accuracy of estimation by training rules of estimation and repeating the run cycles. There are many factors that impact software effort estimates such as team size, concurrency, intensity, fragmentation, software complexity, computer platform and different site characteristics in case of software development.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

The above formula is used for evaluating its performance. RMSE is calculated by applying square root to the MSE.

[4]. Muhammad RazaTayyab etal," ***A Machine Learning Based Model for Software Cost Estimation***".

Efforts had been made to present a comprehensive analysis of software cost estimation. The basic goal to conduct this deep study is to evaluate different methods used for predicting the software development cost to increase our understanding of this area of research. Some of the suggested methods that helps to achieve many benefits for improving software cost estimation with respect to effort and cost like 10 fold cross validation are discussed in this paper.

This section describes different machine learning based software cost estimation techniques use in the literature. In this study, we categorize the cost estimation approaches into two domains namely, cost estimation and effort estimation as described in the following sub-sections.

(i) Effort estimation.

(ii) Cost estimation.

SYSTEM ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

The functional requirement of the system defines a function of software system or its components. A function is described as set of inputs, behaviour of a system and output.

- Fast and efficient
- Simple Computation

3.2 NON-FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or functions.

- Response time : Time taken to get the result
- Memory Usage : Total amount of memory used by system.

3.3 SOFTWARE REQUIREMENTS

Operating System : Window
IDE : Jupyter notebook.
Programming Language : Python 3.4

3.4 HARDWARE REQUIREMENTS

RAM : 4 GB
Memory : 500 GB
Processor : Intel core i5
GPU : Nvidia 940M / AMD

3.5 MODULES

- SKLEARN

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit learn contains all the necessary methods to build machine learning tasks. It can be installed by following command.

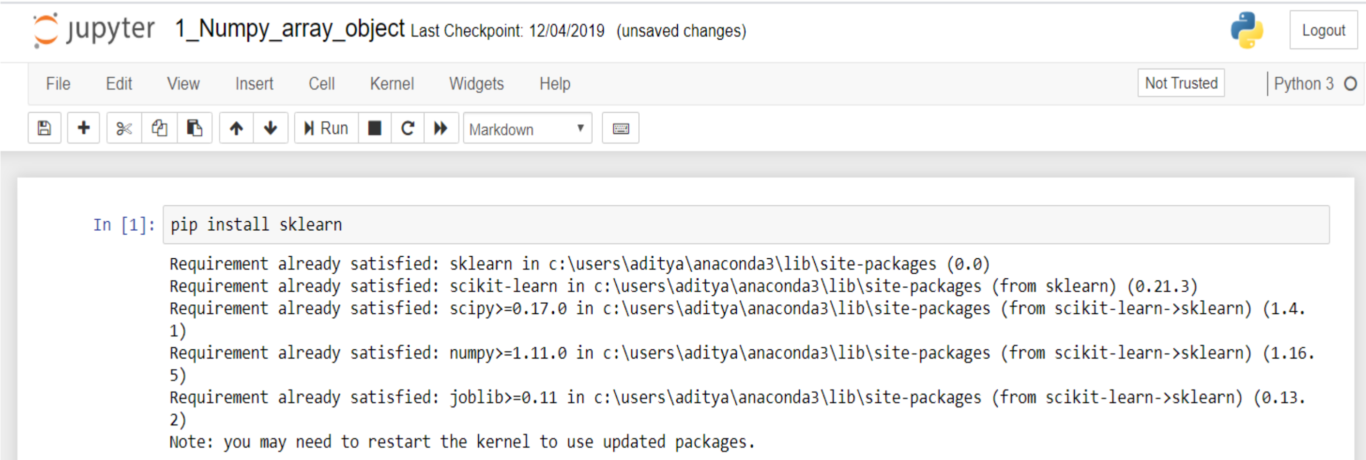


Fig 3.5.1 Sklearn installation command

- NUMPY

NumPy is the fundamental package for scientific computing with Python. It contains among other things such as extension package to Python for multi-dimensional arrays, closer to hardware (efficiency),designed for scientific computation (convenience) and also known as array oriented computing.Following are the some important steps in numpy.

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities.

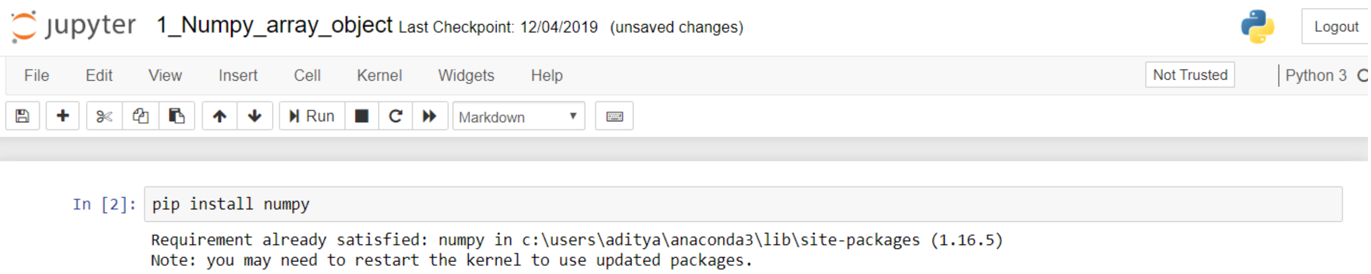


Fig 3.5.2 Numpy installation command.

- PANDAS

  pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. pandas is a Num FOCUS sponsored project. This will help ensure the success of development of pandas as a world-class open-source project and makes it possible to donate to the project. Pandas have data structures such as Dataframes, Series and Panels. Following command is used for this library installation.

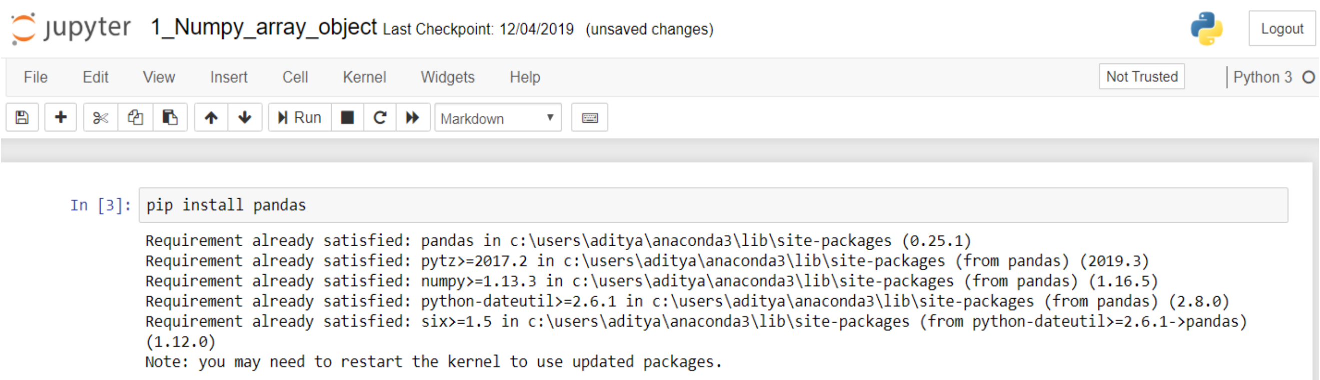


Fig 3.5.3 Pandas installation command

SYSTEM DESIGN

4.1 INTRODUCTION

For this project, only projects from the last decade were used, in order to reflect more recent advancements in software development. In the process of reviewing and selecting data for modelling, firstly dependent variables were chosen. For effort, it was decided to use Normalised Work Effort which presents the total effort required to perform an initiative. In terms of duration, the real elapsed time was obtained by subtracting two variables: Project Elapsed Time and Project Inactive Time (International Software Benchmarking Standards Group, 2013). Next, records which were classified by ISBSG as unreliable, with low quality were removed mainly categories such as C and D from Data Quality Rating. From 125 variables, grouped into 15 categories, only a subset of them was chosen that may influence prediction of effort and duration of software projects at the early stage of lifecycle. As was mentioned in the previous paragraph ISBSG contain a large amount of missing values. Due to a significant number of observations available in the dataset.

Variable	Description	Type	Categories	Role
Industry sector	Organization tye	Nominal	14	Input
Application type	Addressed app. type	Nominal	16	Input
Development type	enhancement	Nominal	3	Input
Development platform	PC ,Mid range	Nominal	4	Input
Language type	Programming language	Nominal	3	Input
Package customization	Check project is having PC or not	Nominal	3	Input
Relative size	Functional points	Nominal	7	Input
Architecture	System architecture	Flag	6	Input
Agile	Agile used	Nominal	2	Input
Used methodology	Development used methodology	Nominal	3	Input
Resource level	Team effort	Nominal	4	Input
Effort	Total project work in months	Continuous	-	Output
Duration	Total project elapsed time	Continuous	-	Output

Table 4.1.1 Selected variables for effort and duration estimation.

4.2 ARCHITECTURE

The following figure shows that how the flow is going to take place while predicting the effort and duration.At first retrival of data from the ISBSG dataset is to be done,after that data preprocessing must be done to avoid noisy data .In this step,we mainly considered the noisy data handling by most frequent item in the column.So,necessary fields are selected.

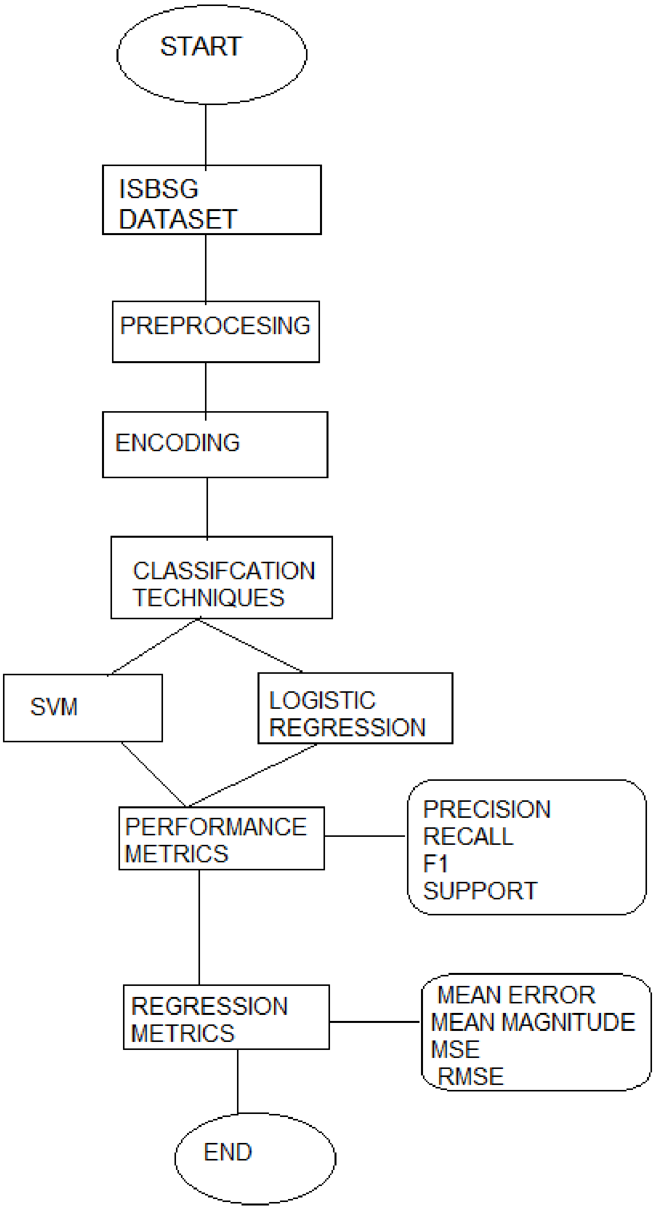


FIG 4.2.2 ARCHITECTURE

4.3 MACHINE LEARNING ALGORITHMS

4.3.1 INTRODUCTION TO SVM

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

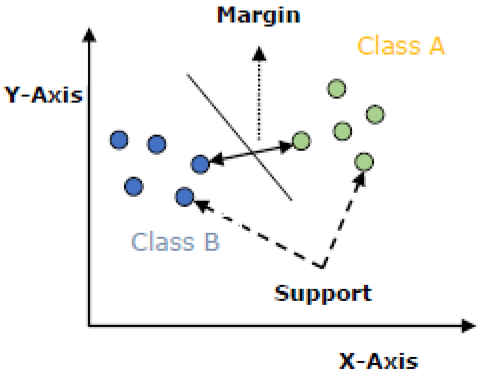


FIG 4.3.1 SVM plane

4.3.2 WORKING OF SVM

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).Following are the steps in Support vector machine.

The followings are important concepts in SVM −

- Support Vectors − Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.
- Hyperplane − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

- Margin − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps −

- First, SVM will generate hyperplanes iteratively that segregates the classes in best way.
- Then, it will choose the hyperplane that separates the classes correctly.

4.3.3 SVM KERNELS

In practice, SVM algorithm is implemented with kernel that transforms an input data space into the required form. SVM uses a technique called the kernel trick in which kernel takes a low dimensional input space and transforms it into a higher dimensional space. In simple words, kernel converts non-separable problems into separable problems by adding more dimensions to it. It makes SVM more powerful, flexible and accurate. The following are some of the types of kernels used by SVM.

4.3.3.1 LINEAR KERNEL

It can be used as a dot product between any two observations. The formula of linear kernel is as shown below along with the parameters used in it −

$$K(x,xi)=sum(x*xi)K(x,xi)=sum(x*xi)$$

From the above formula, we can see that the product between two vectors say x & xi is the sum of the multiplication of each pair of input values.

4.3.3.2 POLYNOMIAL KERNEL

It is more generalized form of linear kernel and distinguish curved or nonlinear input space. Following is the formula for polynomial kernel −

$$k(X,Xi)=1+sum(X*Xi)^dk(X,Xi)=1+sum(X*Xi)^d$$

Here d is the degree of polynomial, which we need to specify manually in the learning algorithm.

4.3.3.3 RADIAL BASIS FUNCTION (RBF) KERNEL

RBF kernel, mostly used in SVM classification, maps input space in indefinite dimensional space. Following formula explains it mathematically −

K(x,xi)=exp(−gamma∗sum(x−xi^2))K(x,xi)=exp(−gamma∗sum(x−xi^2))

Here, *gamma* ranges from 0 to 1. We need to manually specify it in the learning algorithm. A good default value of *gamma* is 0.1.

As we implemented SVM for linearly separable data, we can implement it in Python for the data that is not linearly separable. It can be done by using kernels.

4.4 LOGISTIC REGRESSION

Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable(or output), y, can take only discrete values for given set of features(or inputs), X.

Contrary to popular belief, logistic regression IS a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.

$$g(z) = \frac{1}{1 + e^{-z}}$$

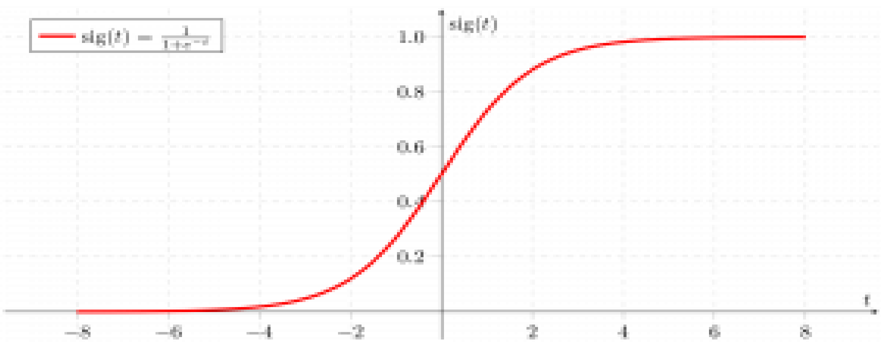


FIG 4.4.1 Logistic regression

Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a very important aspect of Logistic regression and is dependent on the classification problem itself.The decision for the value of the threshold value is majorly affected by the values of precision and recall. Ideally, we want both precision and recall to be 1, but this seldom is the case. In case of a Precision-Recall tradeoff we use the following arguments to decide upon the threshold value.

1. Low Precision/High Recall : In applications where we want to reduce the number of false negatives without necessarily reducing the number false positives, we choose a decision value which has a low value of Precision or high value of Recall. For example, in a cancer diagnosis application, we do not want any affected patient to be classified as not affected without giving much heed to if the patient is being wrongfully diagnosed with cancer. This is because, the absence of cancer can be detected by further medical diseases but the presence of the disease cannot be detected in an already rejected candidate.

2. High Precision/Low Recall : In applications where we want to reduce the number of false positives without necessarily reducing the number false negatives, we choose a decision value which has a high value of Precision or low value of Recall. For example, if we are classifying customers whether they will react positively or negatively to a personalised advertisement, we want to be absolutely sure that the customer will react positively to the advertisement because otherwise, a negative reaction can cause a loss potential sales from the customer.

Based on the number of categories, Logistic regression can be classified as:

- Binomial : Target variable can have only 2 possible types: “0” or “1” which may represent “win” vs “loss”, “pass” vs “fail”, “dead” vs “alive”, etc.
- Multinomial : Target variable can have 3 or more possible types which are not ordered(i.e. types have no quantitative significance) like “disease A” vs “disease B” vs “disease C”.
- Ordinal : It deals with target variables with ordered categories. For example, a test score can be categorized as:“very poor”, “poor”, “good”, “very good”. Here, each category can be given a score like 0, 1, 2, 3.

4.5 PERFORMANCE METRICS

4.5.1 CONFUSION MATRIX

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made. Most performance measures are computed from the confusion matrix.

CONFUSION MATRIX	CLASS 1 PREDICTED	CLASS 2 PREDICTED
CLASS 1 ACTUAL	TRUE POSITIVE	FALSE NEGATIVE
CLASS2 ACTUAL	FALSE POSITIVE	TRUE NEGATIVE

Table 4.5.1 Confusion matrix

Here,

- Class 1 : Positive
- Class 2 : Negative

Definition of the Terms

- Positive (P) : Observation is positive (for example: is an apple).
- Negative (N) : Observation is not positive (for example: is not an apple).
- True Positive (TP) : Observation is positive, and is predicted to be positive.
- False Negative (FN) : Observation is positive, but is predicted negative.
- True Negative (TN) : Observation is negative, and is predicted to be negative.
- False Positive (FP) : Observation is negative, but is predicted positive.

1. Classification Rate/Accuracy

Classification Rate or Accuracy is given by the relation which is as follows.

$$\text{accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}$$

However, there are problems with accuracy. It assumes equal costs for both kinds of errors. A 99% accuracy can be excellent, good, mediocre, poor or terrible depending upon the problem.

2. Recall

Recall can be defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. High Recall indicates the class is correctly recognized (a small number of FN).Following shows the equation for recall.

$$recall = \frac{tp}{tp + fn}$$

3. Precision

To get the value of precision we divide the total number of correctly classified positive examples by the total number of predicted positive examples. High Precision indicates an example labelled as positive is indeed positive (a small number of FP).

$$precision = \frac{tp}{tp + fp}$$

High recall, low precision: This means that most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.
Low recall, high precision: This shows that we miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP)

4. F-measure

Since we have two measures (Precision and Recall) it helps to have a measurement that represents both of them. We calculate an F-measure which uses Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more.
The F-Measure will always be nearer to the smaller value of Precision or Recall.

$$F - Measure = \frac{2 * recall * precision}{recall + precision}$$

Demonstratation of how to create a confusion matrix on a predicted model. We have to import the confusion matrix module from sklearn library which helps us to generate the confusion matrix.

4.6 REGRESSION METRICS

- MAE (Mean absolute error)

  MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$MAE = \overset{\text{Divide by the total number of data points}}{\boxed{\frac{1}{n}}} \underset{\text{Sum of}}{\Sigma} \underbrace{\left| \overset{\text{Actual output value}}{\boxed{y}} - \overset{\text{Predicted output value}}{\boxed{\hat{y}}} \right|}_{\text{The absolute value of the residual}}$$

- MSE (Mean squared error)

  MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate. The MSE is a measure of the quality of an estimator and it is always non-negative, and values closer to zero are better. Following is the equation of MSE.

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \hat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

- RMSE (Root mean square error)

  RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It's the square root of the average of squared differences between prediction and actual observation.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

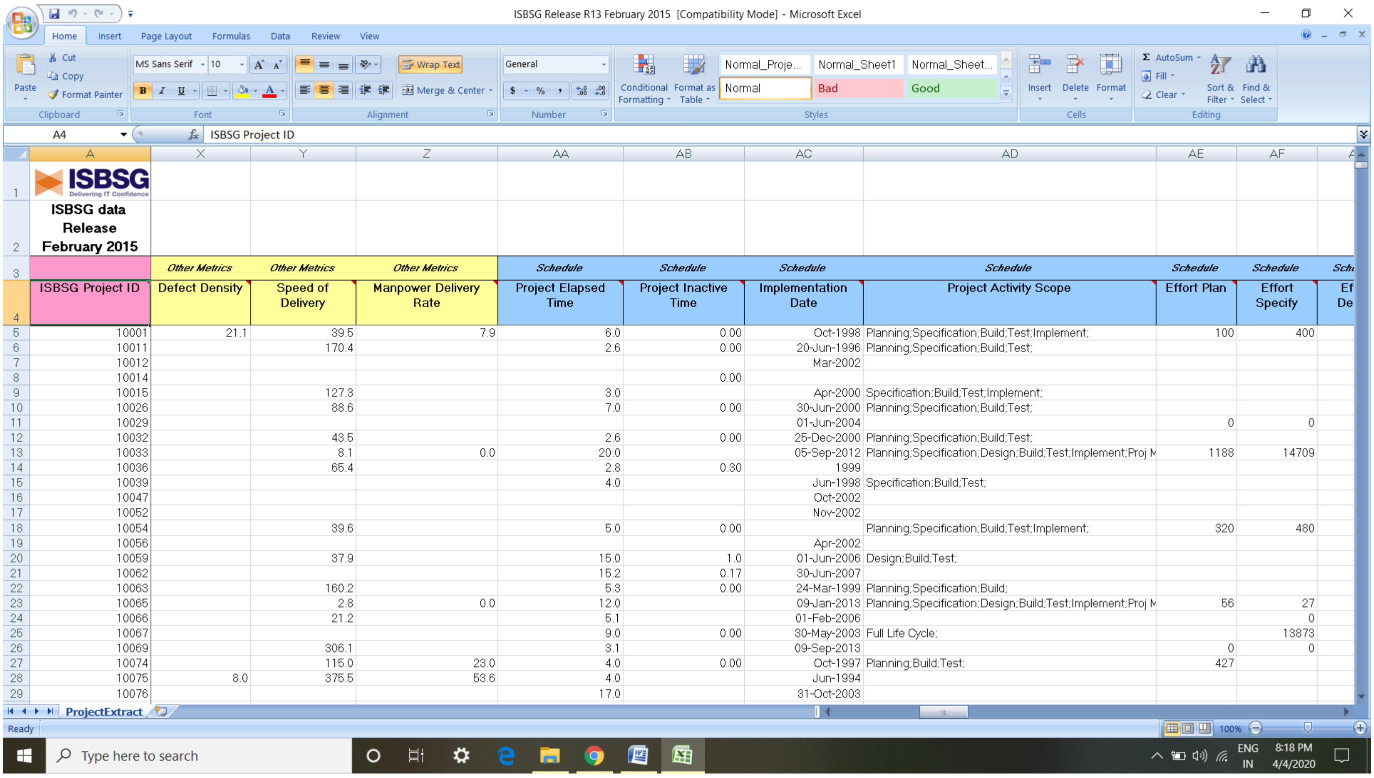
IMPLEMENTATION

5.1 DATA PREPROCESSING

It is a data mining technique that transforms raw data into an understandable format. Raw data (real world data) is always incomplete and that data cannot be sent through a model. That would cause certain errors. That is why we need to preprocess data before sending through a model. Following are the steps that are required to handle missing values for categorical data by most frequent occurance of element in a column.In machine learning, we usually deal with datasets which contains multiple labels in one or more than one columns. These labels can be in the form of words or numbers. To make the data understandable or in human readable form, the training data is often labeled in words.encoding is a technique to solve the problem. Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

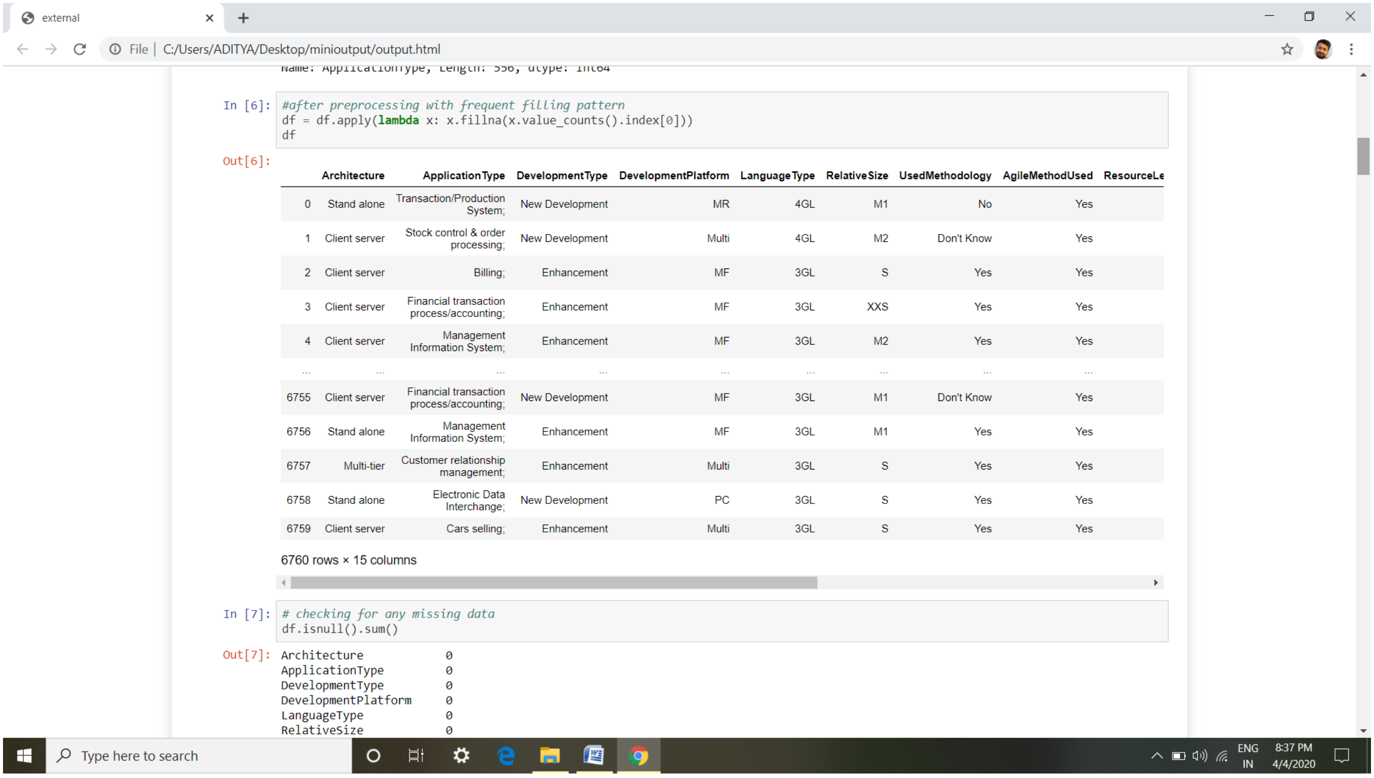
These are the steps for pre-processing of data -

1. Import libraries
2. Read data
3. Checking for missing values
4. Checking for categorical data
5. Standardize the data
6. Data splitting



ISBSG Project ID	Other Metrics: Defect Density	Other Metrics: Speed of Delivery	Other Metrics: Manpower Delivery Rate	Schedule: Project Elapsed Time	Schedule: Project Inactive Time	Schedule: Implementation Date	Schedule: Project Activity Scope	Schedule: Effort Plan	Schedule: Effort Specify
10001	21.1	39.5	7.9	6.0	0.00	Oct-1998	Planning;Specification;Build;Test;Implement;	100	400
10011		170.4		2.6	0.00	20-Jun-1996	Planning;Specification;Build;Test;		
10012						Mar-2002			
10014					0.00				
10015		127.3		3.0		Apr-2000	Specification;Build;Test;Implement;		
10026		88.6		7.0	0.00	30-Jun-2000	Planning;Specification;Build;Test;		
10029						01-Jun-2004		0	0
10032		43.5		2.6	0.00	25-Dec-2000	Planning;Specification;Build;Test;		
10033		8.1	0.0	20.0		05-Sep-2012	Planning;Specification;Design;Build;Test;Implement;Proj M	1188	14709
10036		65.4		2.8	0.30	1999			
10039				4.0		Jun-1998	Specification;Build;Test;		
10047						Oct-2002			
10052						Nov-2002			
10054		39.6		5.0	0.00		Planning;Specification;Build;Test;Implement;	320	480
10056						Apr-2002			
10059		37.9		15.0	1.0	01-Jun-2006	Design;Build;Test;		
10062				15.2	0.17	30-Jun-2007			
10063		160.2		5.3	0.00	24-Mar-1999	Planning;Specification;Build;		
10065		2.8	0.0	12.0		09-Jan-2013	Planning;Specification;Design;Build;Test;Implement;Proj M	56	27
10066		21.2		5.1		01-Feb-2006			0
10067				9.0	0.00	30-May-2003	Full Life Cycle;		13873
10069		306.1		3.1		09-Sep-2013		0	0
10074		115.0	23.0	4.0	0.00	Oct-1997	Planning;Build;Test;	427	
10075	8.0	375.5	53.6	4.0		Jun-1994			
10076				17.0		31-Oct-2003			

Fig 5.1.1 ISBSG dataset



```
In [6]: #after preprocessing with frequent filling pattern
        df = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
        df
```

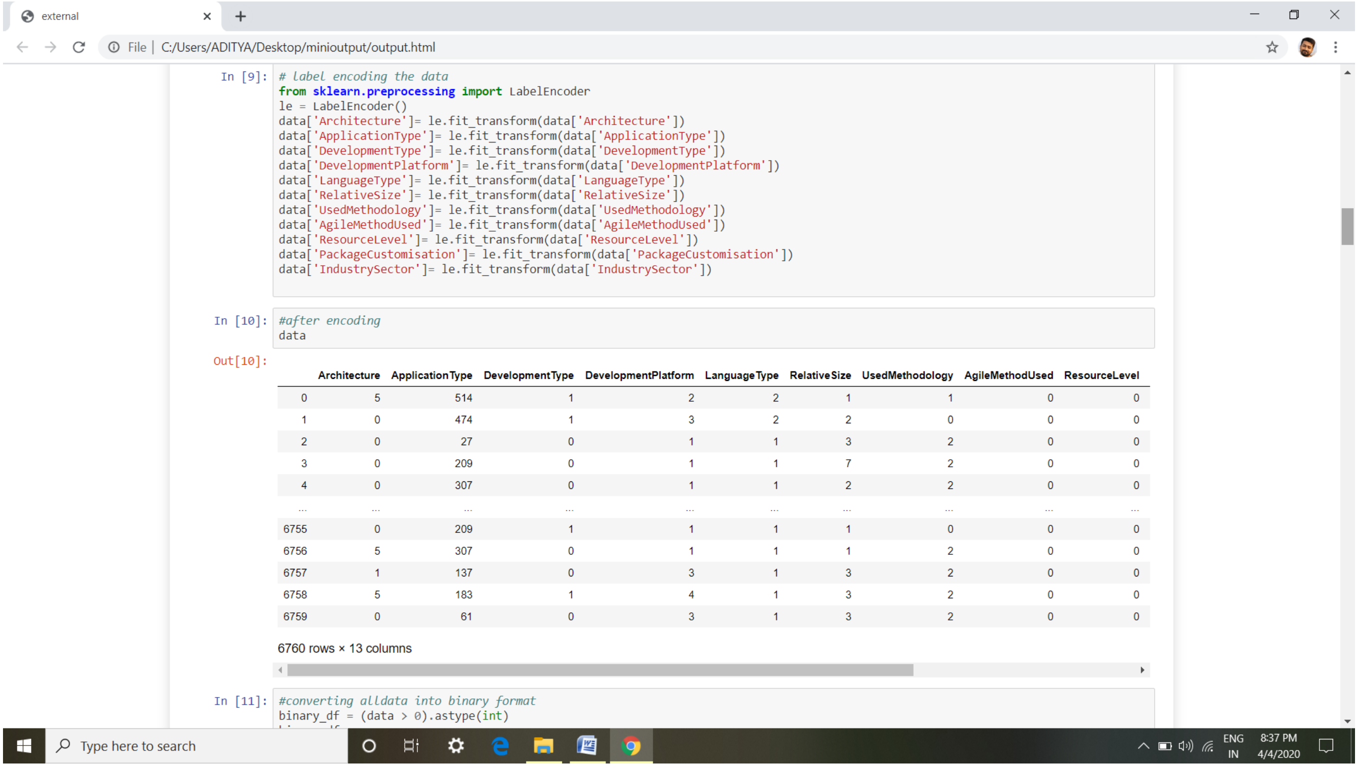
Out[6]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed
0	Stand alone	Transaction/Production System;	New Development	MR	4GL	M1	No	Yes
1	Client server	Stock control & order processing;	New Development	Multi	4GL	M2	Don't Know	Yes
2	Client server	Billing;	Enhancement	MF	3GL	S	Yes	Yes
3	Client server	Financial transaction process/accounting;	Enhancement	MF	3GL	XXS	Yes	Yes
4	Client server	Management Information System;	Enhancement	MF	3GL	M2	Yes	Yes
...	...	...	...	...	...	...	...	...
6755	Client server	Financial transaction process/accounting;	New Development	MF	3GL	M1	Don't Know	Yes
6756	Stand alone	Management Information System;	Enhancement	MF	3GL	M1	Yes	Yes
6757	Multi-tier	Customer relationship management;	Enhancement	Multi	3GL	S	Yes	Yes
6758	Stand alone	Electronic Data Interchange;	New Development	PC	3GL	S	Yes	Yes
6759	Client server	Cars selling;	Enhancement	Multi	3GL	S	Yes	Yes

6760 rows × 15 columns

```
In [7]: # checking for any missing data
        df.isnull().sum()

Out[7]: Architecture           0
        ApplicationType        0
        DevelopmentType        0
        DevelopmentPlatform    0
        LanguageType           0
        RelativeSize           0
```

Fig 5.1.2 Data pre-processing



external

File | C:/Users/ADITYA/Desktop/minioutput/output.html

In [9]:
```python
# label encoding the data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Architecture']= le.fit_transform(data['Architecture'])
data['ApplicationType']= le.fit_transform(data['ApplicationType'])
data['DevelopmentType']= le.fit_transform(data['DevelopmentType'])
data['DevelopmentPlatform']= le.fit_transform(data['DevelopmentPlatform'])
data['LanguageType']= le.fit_transform(data['LanguageType'])
data['RelativeSize']= le.fit_transform(data['RelativeSize'])
data['UsedMethodology']= le.fit_transform(data['UsedMethodology'])
data['AgileMethodUsed']= le.fit_transform(data['AgileMethodUsed'])
data['ResourceLevel']= le.fit_transform(data['ResourceLevel'])
data['PackageCustomisation']= le.fit_transform(data['PackageCustomisation'])
data['IndustrySector']= le.fit_transform(data['IndustrySector'])
```

In [10]:
```python
#after encoding
data
```

Out[10]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel
0	5	514	1	2	2	1	1	0	0
1	0	474	1	3	2	2	0	0	0
2	0	27	0	1	1	3	2	0	0
3	0	209	0	1	1	7	2	0	0
4	0	307	0	1	1	2	2	0	0
...	...	...	...	...	...	...	...	...	...
6755	0	209	1	1	1	1	0	0	0
6756	5	307	0	1	1	1	2	0	0
6757	1	137	0	3	1	3	2	0	0
6758	5	183	1	4	1	3	2	0	0
6759	0	61	0	3	1	3	2	0	0

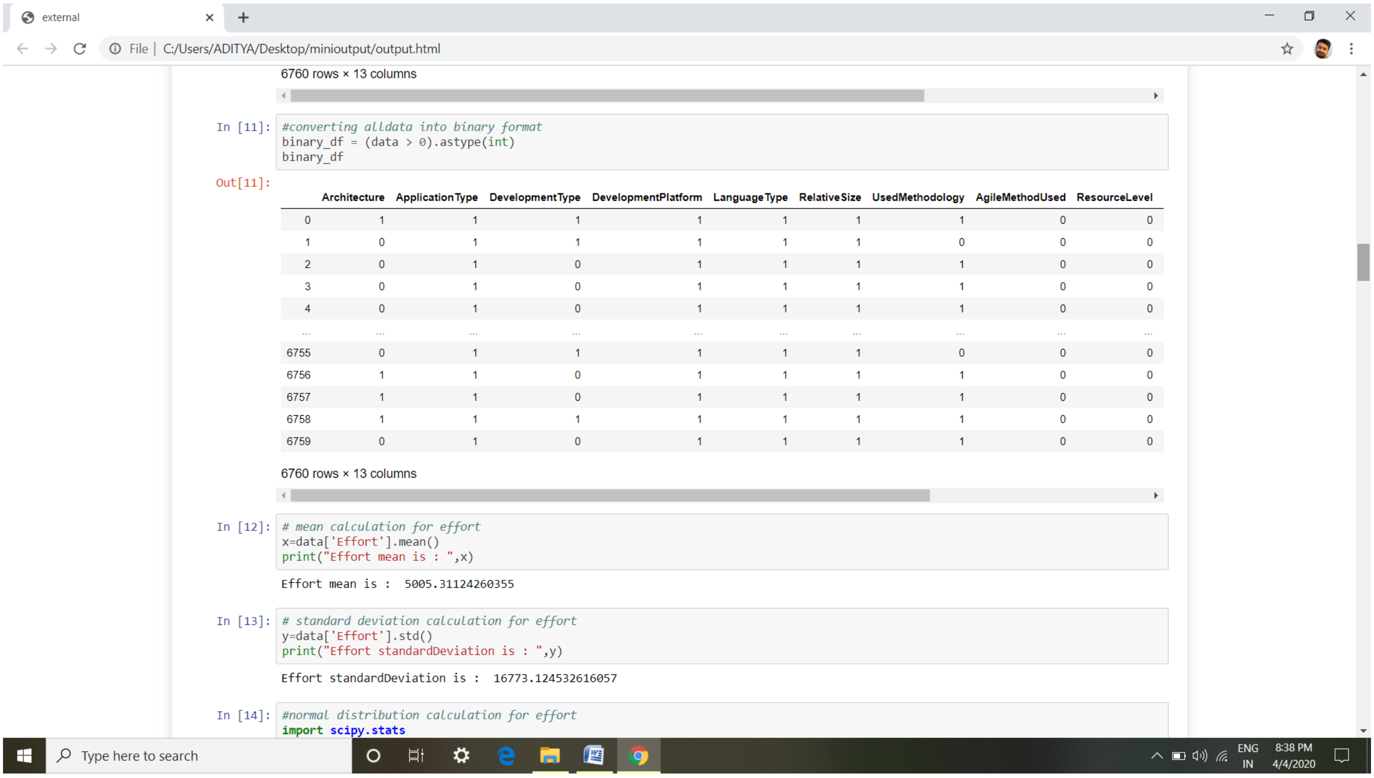
6760 rows × 13 columns

In [11]:
```python
#converting alldata into binary format
binary_df = (data > 0).astype(int)
```

Type here to search

ENG IN 8:37 PM 4/4/2020

Fig 5.1.3 Encoding



external

File | C:/Users/ADITYA/Desktop/minioutput/output.html

6760 rows × 13 columns

In [11]:
```python
#converting alldata into binary format
binary_df = (data > 0).astype(int)
binary_df
```

Out[11]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel
0	1	1	1	1	1	1	1	0	0
1	0	1	1	1	1	1	0	0	0
2	0	1	0	1	1	1	1	0	0
3	0	1	0	1	1	1	1	0	0
4	0	1	0	1	1	1	1	0	0
...	...	...	...	...	...	...	...	...	...
6755	0	1	1	1	1	1	0	0	0
6756	1	1	0	1	1	1	1	0	0
6757	1	1	0	1	1	1	1	0	0
6758	1	1	1	1	1	1	1	0	0
6759	0	1	0	1	1	1	1	0	0

6760 rows × 13 columns

In [12]:
```python
# mean calculation for effort
x=data['Effort'].mean()
print("Effort mean is : ",x)
```

```
Effort mean is :  5005.31124260355
```

In [13]:
```python
# standard deviation calculation for effort
y=data['Effort'].std()
print("Effort standardDeviation is : ",y)
```

```
Effort standardDeviation is :  16773.124532616057
```

In [14]:
```python
#normal distribution calculation for effort
import scipy.stats
```

Type here to search

ENG IN 8:38 PM 4/4/2020

Fig 5.1.4 Effort mean and std-deviation

```
In [20]: #splitting into trainin and test data
         X = binary_df.iloc[:, 1:12].values
         y = binary_df.iloc[:, 12:14].values

In [21]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,random_state=45)

In [22]: #fitting classifier
         from sklearn.svm import SVC
         svclassifier = SVC(kernel='rbf')
         svclassifier.fit(X_train, y_train)

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning: A column-vector y was passe
d when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change from
'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid t
his warning.
  "avoid this warning.", FutureWarning)

Out[22]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
             kernel='rbf', max_iter=-1, probability=False, random_state=None,
             shrinking=True, tol=0.001, verbose=False)

In [23]: y_pred = svclassifier.predict(X_test)

In [24]: from sklearn.metrics import classification_report, confusion_matrix
         from sklearn.metrics import accuracy_score

         print("Accuracy of svm : ",accuracy_score(y_test, y_pred))

         print("confusion matrix \n",
               confusion_matrix (y_test, y_pred))

         Accuracy of svm :  0.8594674556213018
         confusion matrix
          [[   0  190]
          [    0 1162]]

In [25]: print("classification report \n")
```

Fig 5.1.5 SVM

```
In [42]: #logistic regression implementation
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score,confusion_matrix

         clf=LogisticRegression()
         clf.fit(X_train,y_train)
         y_pred=clf.predict(X_test)
         print("Accuracy is : ",accuracy_score(y_test,y_pred))
         print("confusion matrix \n")
         print(confusion_matrix(y_test,y_pred))

         Accuracy is :  0.8579881656804734
         confusion matrix

         [[   0  192]
          [   0 1160]]

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning: A column-vector y was passe
d when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

In [43]: #after cross validation
         result=cross_validate(clf,X_train,y_train,scoring=scoring,cv=10)
         display_result(result)

         TP:  [474 474 473 473 473 473 473 473 473 473]
         TN:  [0 0 0 0 0 0 0 0 0 0]
         FN:  [0 0 0 0 0 0 0 0 0 0]
         FP:  [68 68 68 68 68 68 67 67 67 67]
         Accuracy :  [0.875 0.875 0.874 0.874 0.874 0.874 0.874 0.876 0.876 0.876]

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning: A column-vector y was passe
d when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning: A column-vector y was passe
```

Fig 5.1.6 Logistic Regression

5.2 COCOMO MODEL FOR EFFORT AND DURATION

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Boehm. The model uses a basic regression formula with parameters that are derived from historical project data and current project characteristics. COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes (Cost Drivers). Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases. Following are the values associated with software estimation of project.

5.2.1 BASIC COCOMO

- Basic COCOMO computes software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of source lines of code (SLOC).COCOMO applies to three classes of software projects:
  - Organic projects - "small" teams with "good" experience working with "less than rigid" requirements
  - Semi-detached projects - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
  - Embedded projects - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects.(hardware, software, operational, ...)

The basic COCOMO equations take the form

Effort Applied (E) = a (KLOC)b [man-months]

Development Time (D) = c (Effort Applied)d [months]

People required (P) = Effort Applied / Development Time [count]

where, KLOC is the estimated number of delivered lines (expressed in thousands) of code for project. The coefficients a , b , c and d are given in the following table.

SOFTWARE PROJECT	a	b	C	d
ORGANIC	3.2	1.05	2.5	0.38
SEMI-DETACHED	3.0	1.12	2.5	0.35
EMBEDDED	2.8	1.20	2.5	0.32

Table 5.2.1 cocomo table

5.3 OUTPUT PROGRAMS

START OF PROJECT

In [2]:

```
#import necessary libraries

import numpy as np
import pandas as pd
```

In [3]:

```
#retrive dataset information

df=pd.read_csv('mini.csv')
df
```

Out[3]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel	PackageCustomisation	IndustrySector	Effort	Projectelaspedtime	Projectinactivetime	Duration
0	Stand alone	Transaction/Production System;	New Development	MR	4GL	M1	No	NaN	1.0	No	Service Industry	1850.0	6.0	0.0	6.0
1	NaN	Stock control & order processing;	New Development	Multi	4GL	M2	Don't Know	NaN	1.0	No	Construction	856.0	2.6	0.0	2.6
2	NaN	Billing;	Enhancement	NaN	3GL	S	Yes	NaN	1.0	NaN	Wholesale & Retail	1100.0	NaN	NaN	0.0
3	Client server	NaN	Enhancement	NaN	NaN	XXS	NaN	NaN	1.0	NaN	NaN	28.0	NaN	0.0	0.0
4	Client server	Management Information System;	Enhancement	MF	3GL	M2	Yes	NaN	1.0	No	Wholesale & Retail	23913.0	3.0	NaN	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6755	NaN	NaN	New Development	NaN	NaN	M1	Don't Know	NaN	NaN	Don't Know	Banking	960.0	2.0	0.0	2.0

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel	PackageCustomisation	IndustrySector	Effort	Projectelaspedtime	Projectinactivetime	Duration
6756	Stand alone	Management Information System;	Enhancement	MF	3GL	M1	Yes	NaN	NaN	No	Banking	2312.0	10.0	NaN	10.0
6757	Multi-tier	Customer relationship management;	Enhancement	Multi	3GL	S	NaN	NaN	NaN	NaN	Medical & Health Care	57.0	4.3	NaN	4.3
6758	Stand alone	Electronic Data Interchange;	New Development	PC	3GL	S	Yes	NaN	NaN	No	Electronics & Computers	80.0	1.0	0.0	1.0
6759	Client server	Cars selling;	Enhancement	Multi	3GL	S	Yes	NaN	NaN	NaN	Communication	1449.0	6.0	NaN	6.0

6760 rows × 15 columns

In [4]:

```
# check particular column frequent occurance

df['Architecture'].value_counts()
```

Out[4]:

```
Client server                        1371
Stand alone                          1295
Multi-tier                            382
Multi-tier / Client server            275
Multi-tier with web public interface  164
Multi-tier with web interface          25
Stand-alone                            14
Name: Architecture, dtype: int64
```

In [5]:

```
# consider the most frequent occurance

df['ApplicationType'].value_counts()
```

Out[5]:

```
Financial transaction process/accounting;          987
Transaction/Production System;                     497
not recorded;                                      477
Management Information System;                     375
relatively complex application;                    154
                                                   ...
```

```
Forecastselling;                                            1
Diagnostic distribution  management;                        1
Promotions;                                                 1
Document management;Image, video or sound processing;       1
Data Provisioning;                                          1
Name: ApplicationType, Length: 556, dtype: int64
```

In [6]:

```
#after preprocessing with frequent filling pattern

df = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
df
```

Out[6]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel	PackageCustomisation	IndustrySector	Effort	Projectelaspedtime	Projectinactivetime	Duration
0	Stand alone	Transaction/Production System;	New Development	MR	4GL	M1	No	Yes	1.0	No	Service Industry	1850.0	6.0	0.0	6.0
1	Client server	Stock control & order processing;	New Development	Multi	4GL	M2	Don't Know	Yes	1.0	No	Construction	856.0	2.6	0.0	2.6
2	Client server	Billing;	Enhancement	MF	3GL	S	Yes	Yes	1.0	No	Wholesale & Retail	1100.0	3.0	0.0	0.0
3	Client server	Financial transaction process/accounting;	Enhancement	MF	3GL	XXS	Yes	Yes	1.0	No	Communication	28.0	3.0	0.0	0.0
4	Client server	Management Information System;	Enhancement	MF	3GL	M2	Yes	Yes	1.0	No	Wholesale & Retail	23913.0	3.0	0.0	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6755	Client server	Financial transaction process/accounting;	New Development	MF	3GL	M1	Don't Know	Yes	1.0	Don't Know	Banking	960.0	2.0	0.0	2.0
6	Stan	Manage	Enhanc	MF	3GL	M1	Yes	Yes	1.0	No	Banki	23	10.0	0.0	10.

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel	PackageCustomisation	IndustrySector	Effort	Projectelaspedtime	Projectinactivetime	Duration
756	Stand alone	Management Information System;	Enhancement								...ng	12.0			...0
6757	Multi-tier	Customer relationship management;	Enhancement	Multi	3GL	S	Yes	Yes	1.0	No	Medical & Health Care	57.0	4.3	0.0	4.3
6758	Stand alone	Electronic Data Interchange;	New Development	PC	3GL	S	Yes	Yes	1.0	No	Electronics & Computers	80.0	1.0	0.0	1.0
6759	Client server	Cars selling;	Enhancement	Multi	3GL	S	Yes	Yes	1.0	No	Communication	1449.0	6.0	0.0	6.0

6760 rows × 15 columns

In [7]:

```
# checking for any missing data(noisy data)

df.isnull().sum()
```

Out[7]:

```
Architecture             0
ApplicationType          0
DevelopmentType          0
DevelopmentPlatform      0
LanguageType             0
RelativeSize             0
UsedMethodology          0
AgileMethodUsed          0
ResourceLevel            0
PackageCustomisation     0
IndustrySector           0
Effort                   0
Projectelaspedtime       0
Projectinactivetime      0
Duration                 0
dtype: int64
```

In [8]:

```
#dropping unwanted columns

data=df.drop(['Projectelaspedtime','Projectinactivetime'],axis=1)
data
```

Out[8]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel	PackageCustomisation	IndustrySector	Effort	Duration
0	Stand alone	Transaction/Production System;	New Development	MR	4GL	M1	No	Yes	1.0	No	Service Industry	1850.0	6.0
1	Client server	Stock control & order processing;	New Development	Multi	4GL	M2	Don't Know	Yes	1.0	No	Construction	856.0	2.6
2	Client server	Billing;	Enhancement	MF	3GL	S	Yes	Yes	1.0	No	Wholesale & Retail	1100.0	0.0
3	Client server	Financial transaction process/accounting;	Enhancement	MF	3GL	XXS	Yes	Yes	1.0	No	Communication	28.0	0.0
4	Client server	Management Information System;	Enhancement	MF	3GL	M2	Yes	Yes	1.0	No	Wholesale & Retail	23913.0	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
6755	Client server	Financial transaction process/accounting;	New Development	MF	3GL	M1	Don't Know	Yes	1.0	Don't Know	Banking	960.0	2.0
6756	Stand alone	Management Information System;	Enhancement	MF	3GL	M1	Yes	Yes	1.0	No	Banking	2312.0	10.0
6757	Multi-tier	Customer relationship management;	Enhancement	Multi	3GL	S	Yes	Yes	1.0	No	Medical & Health Care	57.0	4.3

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel	PackageCustomisation	IndustrySector	Effort	Duration
6758	Stand alone	Electronic Data Interchange;	New Development	PC	3GL	S	Yes	Yes	1.0	No	Electronics & Computers	80.0	1.0
6759	Client server	Cars selling;	Enhancement	Multi	3GL	S	Yes	Yes	1.0	No	Communication	1449.0	6.0

6760 rows × 13 columns

In [9]:

```
# label encoding the data

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Architecture']= le.fit_transform(data['Architecture'])
data['ApplicationType']= le.fit_transform(data['ApplicationType'])
data['DevelopmentType']= le.fit_transform(data['DevelopmentType'])
data['DevelopmentPlatform']= le.fit_transform(data['DevelopmentPlatform'])
data['LanguageType']= le.fit_transform(data['LanguageType'])
data['RelativeSize']= le.fit_transform(data['RelativeSize'])
data['UsedMethodology']= le.fit_transform(data['UsedMethodology'])
data['AgileMethodUsed']= le.fit_transform(data['AgileMethodUsed'])
data['ResourceLevel']= le.fit_transform(data['ResourceLevel'])
data['PackageCustomisation']= le.fit_transform(data['PackageCustomisation'])
data['IndustrySector']= le.fit_transform(data['IndustrySector'])
```

In [10]:

```
#after encoding

data
```

Out[10]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel	PackageCustomisation	IndustrySector	Effort	Duration
0	5	514	1	2	2	1	1	0	0	1	10	1850.0	6.0
1	0	474	1	3	2	2	0	0	0	1	2	856.0	2.6

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel	PackageCustomisation	IndustrySector	Effort	Duration
2	0	27	0	1	1	3	2	0	0	1	12	1100.0	0.0
3	0	209	0	1	1	7	2	0	0	1	1	28.0	0.0
4	0	307	0	1	1	2	2	0	0	1	12	23913.0	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
6755	0	209	1	1	1	1	0	0	0	0	0	960.0	2.0
6756	5	307	0	1	1	1	2	0	0	1	0	2312.0	10.0
6757	1	137	0	3	1	3	2	0	0	1	8	57.0	4.3
6758	5	183	1	4	1	3	2	0	0	1	3	80.0	1.0
6759	0	61	0	3	1	3	2	0	0	1	1	1449.0	6.0

6760 rows × 13 columns

In [11]:

```
#converting alldata into binary format

binary_df = (data > 0).astype(int)
binary_df
```

Out[11]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel	PackageCustomisation	IndustrySector	Effort	Duration
0	1	1	1	1	1	1	1	0	0	1	1	1	1
1	0	1	1	1	1	1	0	0	0	1	1	1	1
2	0	1	0	1	1	1	1	0	0	1	1	1	0

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	RelativeSize	UsedMethodology	AgileMethodUsed	ResourceLevel	PackageCustomisation	IndustrySector	Effort	Duration
3	0	1	0	1	1	1	1	0	0	1	1	1	0
4	0	1	0	1	1	1	1	0	0	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
6755	0	1	1	1	1	1	0	0	0	0	0	1	1
6756	1	1	0	1	1	1	1	0	0	1	0	1	1
6757	1	1	0	1	1	1	1	0	0	1	1	1	1
6758	1	1	1	1	1	1	1	0	0	1	1	1	1
6759	0	1	0	1	1	1	1	0	0	1	1	1	1

6760 rows × 13 columns

In [12]:

```
# mean calculation for effort

x=data['Effort'].mean()
print("Effort mean is : ",x)
```

Effort mean is : 5005.31124260355

In [13]:

```
# standard deviation calculation for effort
y=data['Effort'].std()
print("Effort standardDeviation is : ",y)
```

Effort standardDeviation is : 16773.124532616057

In [14]:

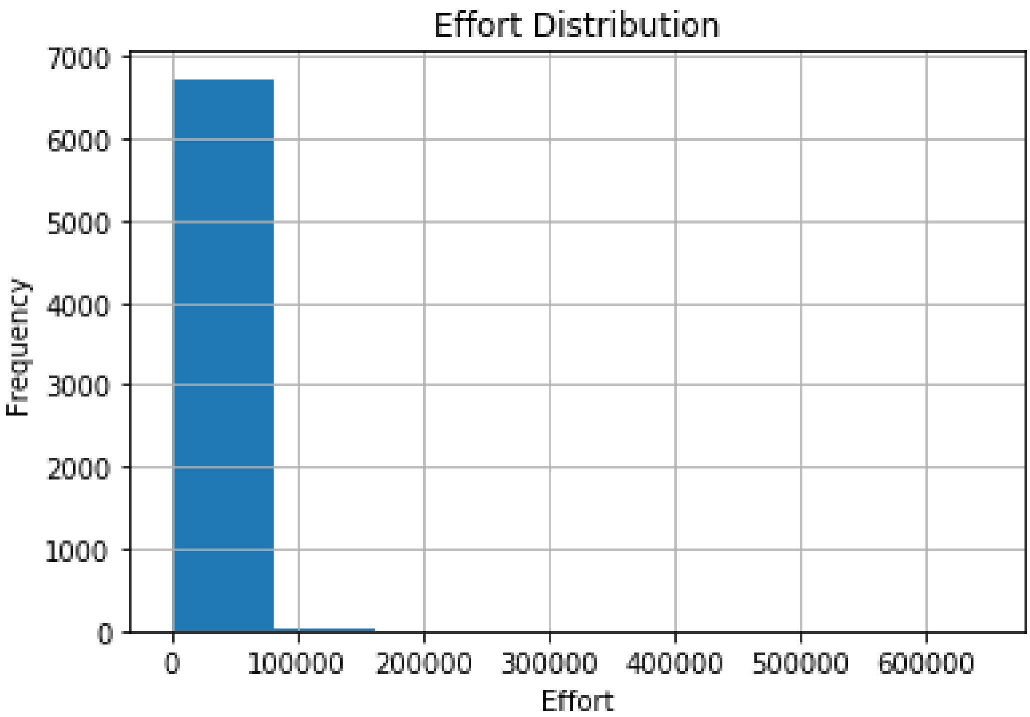
```
#normal distribution calculation for effort
import scipy.stats
z=scipy.stats.norm(5005.31, 16773.12).pdf(98)
print("Effort normal distribution is : ",z)
```

Effort normal distribution is : 2.278814781804906e-05

In [34]:

```
#plotting effort data

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.figure(figsize=[5,8])
data.hist(column='Effort',bins=8)
plt.xlabel('Effort')
plt.ylabel('Frequency')
plt.title('Effort Distribution')
plt.show()
```

<Figure size 360x576 with 0 Axes>



In [16]:

```
# mean calculation for duration

w=data['Duration'].mean()
print("Duration mean is : ",w)
```

Duration mean is : 7.048316568047344

In [17]:

```
# standard deviation calculation for duration

m=data['Duration'].std()
print("Duration standard deviation is : ",m)
```

Duration standard deviation is : 6.978858870981555

In [18]:

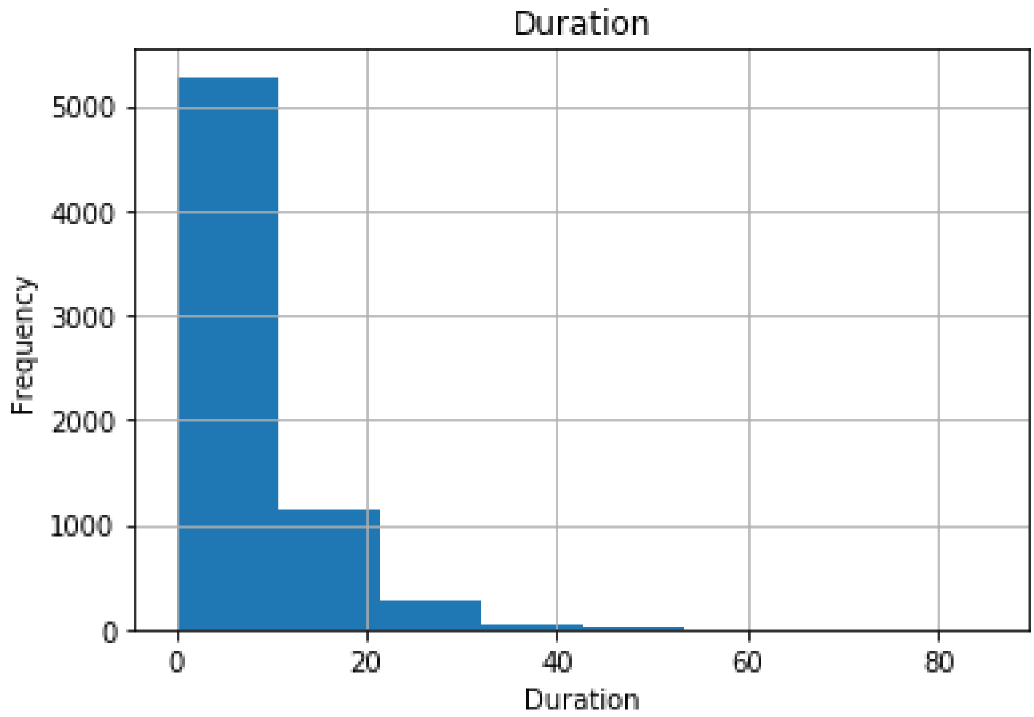
```
#normal distribution calculation for duration

import scipy.stats
n=scipy.stats.norm(7.04, 6.97).pdf(100)
print("Duration normal distribution is : ",n)
```

```
Duration normal distribution is :  1.3538352346476263e-40
```

In [19]:

```
#plotting Duration data

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.figure(figsize=[10,8])
data.hist(column='Duration',bins=8)
plt.xlabel('Duration')
plt.ylabel('Frequency')
plt.title('Duration')
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```



In [20]:

```
#splitting into training and test data

X = binary_df.iloc[:, 1:12].values
y = binary_df.iloc[:, 12:14].values
```

In [21]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 
0.20,random_state=45)
```

In [22]:

```
#fitting classifier

from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)
```

```
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: 
DataConversionWarning: A column-vector y was passed when a 1d array was expected. 
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: 
The default value of gamma will change from 'auto' to 'scale' in version 0.22 to 
account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to 
avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[22]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [23]:

```
y_pred = svclassifier.predict(X_test)
```

In [24]:

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score

print("Accuracy of svm : ",accuracy_score(y_test, y_pred))

print("confusion matrix \n",
      confusion_matrix (y_test, y_pred))
```

```
Accuracy of svm :  0.8594674556213018

confusion matrix 

 [[   0  190]
 [   0 1162]]
```

In [25]:

```
print("classification report \n")
print(classification_report(y_test, y_pred))
classification report


        precision   recall   f1-score   support

0         0.00       0.00      0.00       190
1         0.86       1.00      0.92      1162

accuracy                              0.86      1352
macro avg        0.43       0.50      0.46      1352
weighted avg     0.74       0.86      0.79      1352

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1437:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [26]:

```
#regression metrics implementation

from sklearn.metrics import mean_absolute_error
mae=mean_absolute_error(y_test, y_pred)
print("MAE is %.2f " %mae)

MAE is 0.14
```

In [27]:

```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_test, y_pred)
print("MSE is %.2f"%mse)

MSE is 0.14
```

In [28]:

```
from sklearn.metrics import max_error
me=max_error(y_test,y_pred)
print("ME is %.2f"%me)

ME is 1.00
```

In [29]:

```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse = sqrt(mean_squared_error(y_test, y_pred))
print("RMSE is %.2f" %rmse)

RMSE is 0.37
```

In [30]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.20,random_state=0)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(5408, 11)
(1352, 11)
(5408, 1)
(1352, 1)
```

In [59]:

```
#cross validation

X = binary_df.iloc[:, 1:12].values
y = binary_df.iloc[:, 12:14].values
```

In [60]:

```
from sklearn.model_selection import cross_validate
from sklearn.metrics import make_scorer

def tn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 0]
def fp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 1]
def fn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 0]
def tp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 1]
def acc(y_true, y_pred): return round(accuracy_score(y_true, y_pred),3)

#cross validation purpose

scoring = {'accuracy': make_scorer(accuracy_score),'prec': 'precision'}
scoring = {'tp': make_scorer(tp), 'tn': make_scorer(tn),
           'fp': make_scorer(fp), 'fn': make_scorer(fn),
          'accuracy': make_scorer(acc)}



def display_result(result):
    print("TP: ",result['test_tp'])
    print("TN: ",result['test_tn'])
    print("FN: ",result['test_fn'])
    print("FP: ",result['test_fp'])
    print("Accuracy : ",result['test_accuracy'])
```

In [61]:

```
result=cross_validate(clf,X_train,y_train,scoring=scoring,cv=3)
display_result(result)
```

```
TP:  [1578 1577 1577]
TN:  [0 0 0]
FN:  [0 0 0]
FP:  [226 225 225]

Accuracy :  [0.875 0.875 0.875]

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
  FutureWarning)

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [42]:

```
#logistic regression implementation

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix

clf=LogisticRegression()
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print("Accuracy is : ",accuracy_score(y_test,y_pred))
print("confusion matrix \n")
print(confusion_matrix(y_test,y_pred))
```

```
Accuracy is :  0.8579881656804734

confusion matrix

[[   0  192]
 [   0 1160]]
```

```
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
```

In [43]:

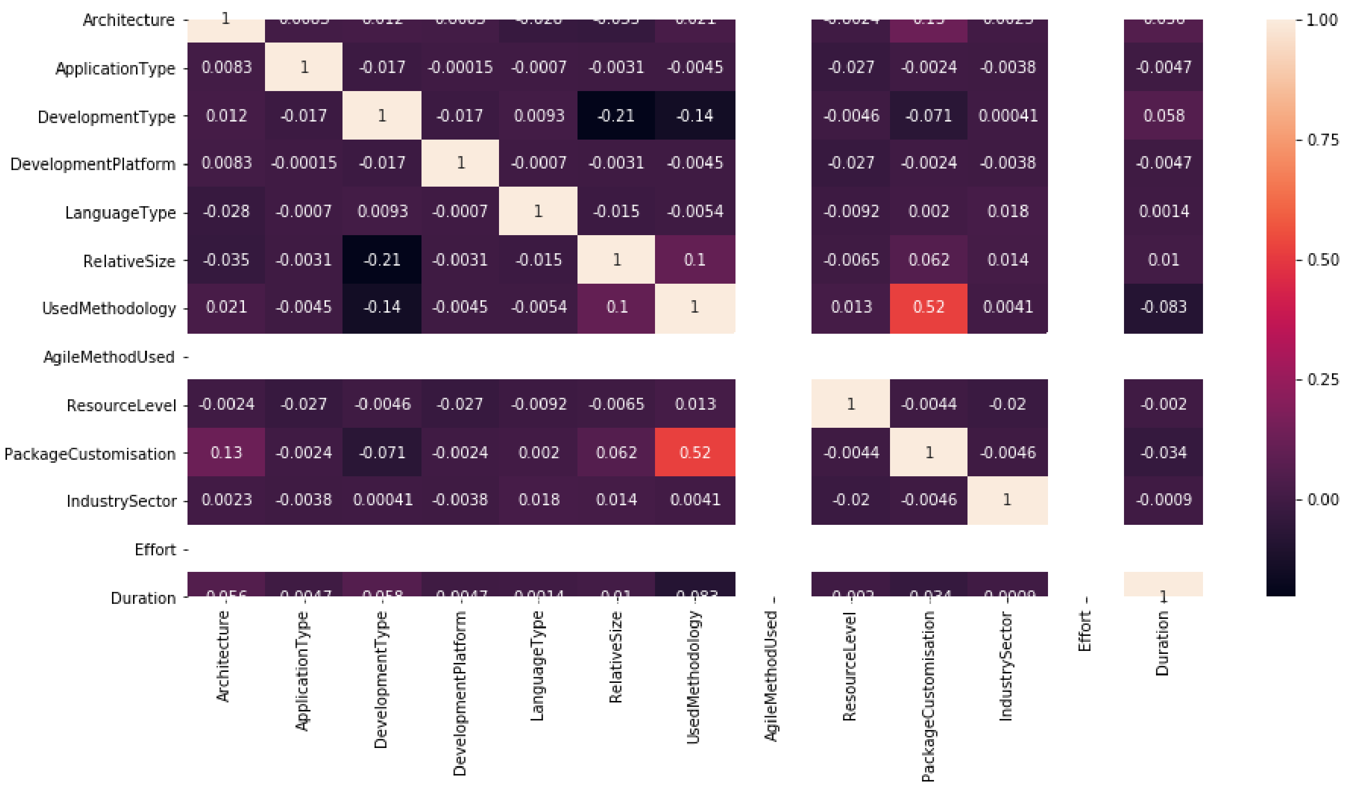
```
#after cross validation

result=cross_validate(clf,X_train,y_train,scoring=scoring,cv=10)
display_result(result)
```

```
TP:  [474 474 473 473 473 473 473 473 473 473]
TN:  [0  0  0  0  0  0  0  0  0  0]
FN:  [0  0  0  0  0  0  0  0  0  0]
FP:  [68  68  68  68  68  68  67  67  67  67]

Accuracy :  [0.875 0.875 0.874 0.874 0.874 0.874 0.876 0.876 0.876 0.876]
```

```
#heat map representation

import seaborn as sns
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(binary_df.corr(), annot=True)
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x206056b2d08>
```



In [45]:

```python
# calculating effort and duration

import pandas as pd
```

In [46]:

```python
df=pd.read_csv('loc.csv')
df
```

Out[46]:

	LOC
0	1,250
1	47,250
2	6,800
3	5,200
4	1,516
...	...
517	33,500
518	1,760
519	10,080
520	453,824
521	300,000

522 rows × 1 columns

In [47]:

```python
#convert to numeric

df['LOC'] = pd.to_numeric(df['LOC'], errors='coerce')
df
```

Out[47]:

	LOC
0	NaN

	LOC
1	NaN
2	NaN
3	NaN
4	NaN
...	...
517	NaN
518	NaN
519	NaN
520	NaN
521	NaN

522 rows × 1 columns

In [48]:

```
df.describe()
```

Out[48]:

	LOC
count	29.000000
mean	469.724138
std	277.166775
min	96.000000
25%	215.000000
50%	410.000000
75%	702.000000

LOC

max 973.000000

In [49]:

```
#calculate mean for LOC

x=df['LOC'].mean()
print(int(x))
```

469

In [50]:

```
# effort calculation in person months

a=3.2
b=1.05
KLOC=469
effort=int(a*(KLOC)**b)
print("Effort","=",effort ,"(Person Months)")
```

Effort = 2041 (Person Months)

In [51]:

```
#duration calculation in months

c=2.5
d=0.38
Effort=2041
Duration = int(c*(Effort)**d)
print("Duration","=",Duration ,"(Months)")
```

Duration = 45 (Months)

In [52]:

```
#persons required to complete project
effort=2041
duration=45
PersonsRequired=effort//duration
print("Persons Required", "=", PersonsRequired)
```

Persons Required = 45

In [53]:

```
# effort calculation in person months
a=3
b=1.12
KLOC=469
effort=int(a*(KLOC)**b)
print("Effort","=",effort ,"(Person Months)")
```

Effort = 2943 (Person Months)

In [54]:

```
#duration calculation in months

c=2.5
d=0.35
Effort=2943
Duration = int(c*(Effort)**d)
print("Duration","=",Duration ,"(Months)")
```

Duration = 40 (Months)

In [55]:

```
#persons required to complete project

effort=2943
duration=40
PersonsRequired=effort//duration
print("Persons Required", "=", PersonsRequired)
```

Persons Required = 73

In [56]:

```
# effort calculation in person months

a=2.8
b=1.20
KLOC=469
effort=int(a*(KLOC)**b)
print("Effort","=",effort ,"(Person Months)")
```

Effort = 4493 (Person Months)

In [57]:

```
#duration calculation in months
c=2.5
d=0.32
Effort=4493
Duration = int(c*(Effort)**d)
print("Duration","=",Duration ,"(Months)")
```

Duration = 36 (Months)

In [58]:

```
#persons required to complete project
effort=4493
duration=36
PersonsRequired=effort//duration
print("Persons Required", "=", PersonsRequired)
```

Persons Required = 124

In [ ]:

END OF PROJECT

RESULTS AND DISCUSSION

The implementation of machine learning models for software effort and duration estimation requires numerous prerequisites. Depending on diversity of initiatives , their size and nature, the number of required completed projects may vary, but 40– 60 should be sufficient. This may not be achievable for small organisations, therefore the models are more intended for medium and large size organisations where incorrect estimation can lead to significant implications. Additionally, prior to implementation, a cost benefit analysis should be performed, which for companies that conduct mostly small projects may have negative results, therefore, traditional methods based on expert knowledge could be sufficient.Following are the results that shows the performance of various ML algorithms and metrics.

PRECISION	RECALL	F1-SCORE	SUPPORT
0.77	0.89	0.86	1352
0.43	0.50	0.46	1352
0.74	0.86	0.79	1352

Table 6.1 Performance metrics for software effort and duration estimation

REGRESSION METRICS	ERROR RATE
MEAN ABSOLUTE ERROR	0.14
MEAN SQUARED ERROR	0.14
MAX ERROR	1.00
ROOT MEAN SQUARED ERROR	0.37

Table 6.2 Regression metrics for software effort and duration estimation

CLASSIFICATION TECHNIQUES	BEFORE CROSS VALIDATION	AFTER CROSS VALIDATION
SUPPORT VECTOR MACHINE	0.85	0.87
LOGISTIC REGRESSION	0.85	0.88

Table 6.3 Classification accuracy for software effort and duration estimation

SOFTWARE PROJECT	EFFORT A	EFFORT B	DURATION C	DURATION D
ORGANIC	3.2	1.05	2.5	0.38
SEMI-DETACHED	3.0	1.12	2.5	0.35
EMBEDDED	2.8	1.20	2.5	0.32

Table 6.4 Prediction for software Effort and Duration estimation

CONCLUSION

The proposed effort and duration estimation models are intended to serve as a decision support tool for any organisation developing and implementing software systems regardless of the industry sector where incorrect estimation may lead to negative implications.Considering the current dynamic software development environment with agility, prototyping and rapid delivery, software simulation using machine learning algorithms could further enhance project estimation methods and contribute to better resource allocation and utilization will be the future scope.

REFERENCES

[1].Przemyslaw Pospieszny , " ***An effective approach for software project effort and duration estimation with machine learning algorithms*** ", The journal of systems and software, Vol.137,pp.184-196, (2018).

[2]. Jianglin Huang etal**,"** ***An empirical analysis of data preprocessing for machine learning-based software cost estimation***", Information and Software Technology,Vol.67,pp.108-127,(2016).

[3]. Rekha Tripathi et al," ***Machine Learning Methods of Effort Estimation and It's Performance Evaluation Criteria*** ", International Journal of Computer Science and Mobile Computing, Vol.6, pp. 61-67,(2017).

[4]. Muhammad Raza Tayyab etal," ***A Machine Learning Based Model for Software Cost Estimation***" , Springer International Publishing ,Vol.16,(2016).

[5]. R. Saljoughinejad and V. Khatibi, "***A new optimized hybrid model based On COCOMO to increase the accuracy of software cost estimation***," Journal of Advances in Computer Engineering and Technology, vol. 4, pp. 27-40, (2018).

[6]. Farrukh Arslan ," ***A Review of machine learning models for software cost estimation***", Review of Computer Engineering Research,Vol.6, No.2, pp. 64-75,(2019).