**Major Project Report**

**On**

“**Car Resale Price Predictor**”

Submitted in partial fulfillment of the

Requirements for the award of the degree of

**Bachelor of Technology**

**In**

**Computer Science & Engineering**

**By**

**Varun Panchal – 18R21A05H8**

**B Abhishek – 18R21A05C7**

**Jatoth Pavan – 18R21A05E3**

**K Sai Naga Teja – 18R21A05E4**

Under the guidance of

**MS D Divya Priya**

**Associate Professor**

**Department of Computer Science & Engineering**

**2022**



**Department of Computer Science & Engineering**

**CERTIFICATE**

This is to certify that the project entitled **“Car Resale Price Predictor”** has been submitted by **Varun Panchal (18R21A05H8), B Abhishek (18R21A05C7), Jatoth Pavan (18R21A05E3), K Sai Naga Teja (18R21A05E4)** in partial fulfillment of the requirements for the award ofdegree of Bachelor of Technology in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad. The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

**Internal Guide** **Head of the Department**

**External Examiner**

**i**



**Department of Computer Science & Engineering**

**DECLARATION**

We hereby declare that the project entitled **“Car Resale Price Predictor”** is the work done during the periodfrom **March 2021 to June 2021** and is submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Jawaharlal Nehru Technology University, Hyderabad. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

**Varun Panchal 18R21A05H8**

**Jatoth Pavan 18R21A05E3**

**Sai Teja 18R21A05E4**

**B Abhishek 18R21A05C7**

**ii**



**Department of Computer Science & Engineering**

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task  
would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we now have the opportunity to express our guidance for all of them.

First of all, We would like to express our deep gratitude towards our internal guide **MS D. Divya Priya, Assistant Professor, Department of CSE** for her support in the completion ofour dissertation. We wish to express our sincere thanks to **Dr. E. Anu Priya,** HOD, Dept. of CSE and also principal **Dr. K. SRINIVAS RAO** for providing the facilities tocomplete the dissertation.

We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

**Varun Panchal 18R21A05H8**

**Jatoth Pavan 18R21A05E3**

**Sai Teja 18R21A05E4**

**B Abhishek 18R21A05C7**

**iii**



**Department of Computer Science & Engineering**

**ABSTRACT**

The prices of new cars in the industry are fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But due to the increased price of new cars and the incapability of customers to buy new cars due to the lack of funds, used cars sales are on a global increase. There is a need for a used car price prediction system to effectively determine the worthiness of the car using a variety of features. Even though there are websites that offers this service, their prediction method may not be the best. Besides, different models and systems may contribute on predicting power for a used car’s actual market value. It is important to know their actual market value while both buying and selling.

The machine learning techniques used in predicting price of cars are Random Forest, AdaBoost Regression, XGBoost Regression. The process also includes justifications during data cleaning, comparison of the performance of the models and reporting the finding of the study in a professional manner. This will help user to get the price of used car for both buyer and seller without taking help of a third party source or inspection.

**iv**

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure Number** | **Name Of The Figure** | **Page Number** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

v

**INDEX**

**Certificate i**

**Declaration ii**

**Acknowledgement iii**

**Abstract iv**

**List of Figures** **v**

**Chapter 1 1-4**

**Introduction**

1.1Overview 1-2

1.2 Purpose of the project 3-4

**Chapter 2 5**

**Literature Survey**

2.1 Cloud Drive Multiple Downloader 5

2.2 Disadvantages of Existing System 5

**Chapter 3 6-7**

**Proposed System**

3.1 Proposed System 6

3.2 Advantages of Proposed System. 6

3.3 System Requirements 7

3.3.1 Software Requirements 7

3.3.2 Hardware Requirements 7

3.3.3 Implementation Technologies 7

**Chapter 4 8-12**

**System Design**

4.1 Proposed system Architecture 8

4.2 Modules 9-12

4.2.1 Google Drive API 9

4.2.2 OAuth 2.0 10

4.2.3 Google – Auth 11

4.2.4 PyQt5 11

4.2.5 Auto PY to EXE tool 12

vi

**Chapter 5 13-30**

**Implementation**

5.1 Algorithms 13

5.2 Implementation Steps 14

5.3 Source Code 15

**Chapter 7 31-33**

**Results**

**Chapter 8 34**

**Conclusion**

**Future Enhancement 35**

**Reference 36**

**vi**

# CHAPTER 1

# INTRODUCION

* 1. **OVERVIEW**

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

The learning system of a machine learning algorithm is broke down into three parts.

**A Decision Process:** In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labelled or unlabeled, your algorithm will produce an estimate about a pattern in the data.

**An Error Function:** An error function serves to evaluate the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.

**An Model Optimization Process:** If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluate and optimize process, updating weights autonomously until a threshold of accuracy has been met.

Machine learning classifiers fall into three primary categories.

[**Supervised learning**](https://www.ibm.com/cloud/learn/supervised-learning)**,** also known as supervised machine learning, is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately. This occurs as part of the cross validation process to ensure that the model avoids [overfitting](https://www.ibm.com/cloud/learn/overfitting) or [underfitting](https://www.ibm.com/cloud/learn/underfitting). Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox. Some methods used in supervised learning include neural networks, naïve bayes, linear regression, logistic regression, random forest, support vector machine (SVM), and more.

[**Unsupervised learning**](https://www.ibm.com/cloud/learn/unsupervised-learning), also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, image and pattern recognition. It’s also used to reduce the number of features in a model through the process of dimensionality reduction; principal component analysis (PCA) and singular value decomposition (SVD) are two common approaches for this. Other algorithms used in unsupervised learning include neural networks, k-means clustering, probabilistic clustering methods, and more.

**Semi-supervised learning** offers a happy medium between supervised and unsupervised learning. During training, it uses a smaller labeled data set to guide classification and feature extraction from a larger, unlabeled data set. Semi-supervised learning can solve the problem of having not enough labeled data (or not being able to afford to label enough data) to train a supervised learning algorithm.

**1.2 Purpose of the project**

The prices of new cars in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But due to the increased price of new cars and the incapability of customers to buy new cars due to the lack of funds, used cars sales are on a global increase.

There is a need for a used car price prediction system to effectively determine the worthiness of the car using a variety of features. Even though there are websites that offers this service, their prediction method may not be the best. Besides, different models and systems may contribute on predicting power for a used car’s actual market value. It is important to know their actual market value while both buying and selling.

**The Client,** will be able to predict used cars market value can help both buyers and sellers.

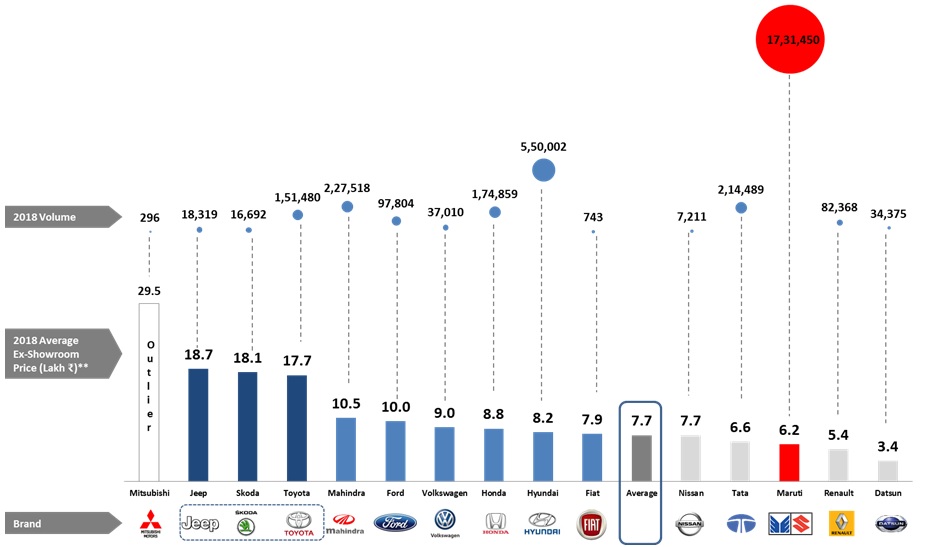
**Used car sellers (dealers):**They are one of the biggest target group that can be interested in results of this study. If used car sellers better understand what makes a car desirable, what the important features are for a used car, then they may consider this knowledge and offer a better service.

**Online pricing services:** There are websites that offers an estimate value of a car. They may have a good prediction model. However, having a second model may help them to give a better prediction to their users. Therefore, the model developed in this study may help online web services that tells a used car’s market value.

**Individuals:** There are lots of individuals who are interested in the used car market at some points in their life because they wanted to sell their car or buy a used car. In this process, it’s a big corner to pay too much or sell less then it’s market value.

**1.3 Motivation**

The automotive industry is composed of a few top global multinational players and several retailers. The multinational players are mainly manufacturers by trade whereas the retail market features players who deal in both new and used vehicles. The used car market has demonstrated a significant growth in value contributing to the larger share of the overall market. The used car market in India accounts for nearly 3.4 million vehicles per year.



**CHAPTER 2**

**2 LITERATURE SURVEY**

We did a background survey regarding the basic ideas of our project and used those ideas for the collection of information like the technological stack, algorithms, and shortcomings of our project which led us to build a better project.

**2.1 Existing System**

**2.1.1 Car Price Prediction Using Machine Learning**

**[IJCSE Research Paper Vol.-7, Issue-5, May 2019, E-ISSN: 2347-2693]**

Upon study on this paper, it has been found that the Machine learning Algorithm which has been used for implementation of model is K-Means Algorithm and Decision Tree Algorithm. The features that have been used to predict the price of a car are Kilometers Traveled, Fiscal Power, Year of registration, Fuel Type. There is no information about the Datasets which has been used for training. Hence, we can assume that the datasets have been taken from the online source.

**2.1.2 Car Price Prediction using Machine Learning Techniques**

**[TEM Journal. Volume 8, Issue 1, ISSN 2217-8309, February 2019]**

Upon study on this paper, it has been found that the Machine Learning Algorithm which has been used for implementation of model is Random Forest Algorithm. The features that have been used to predict the price of a car are Brand, Model, Fuel Type, HorsePower, Year of Manufacture, Miles, Leather, Cruise Control. There is no information about the Datasets which has been used for training. Hence, we can assume that the datasets have been taken from the online source.

**2.1.3 Predicting the Price of Used Cars using Machine Learning Techniques**

**[IJICT Research, ISSN 0974-2239 Volume 4, Number 7 (2014), pp. 753-764]**

Upon study on this paper, it has been found that the Machine learning Algorithm which has been used for implementation of model is Multiple Linear Regression Analysis, K-Nearest Neighbours (kNN), Decision Trees, Naïve Bayes. The features that have been used to predict the price of a car are Company name, Cylinder Volume, Year, Mileage/KM. . There is no information about the Datasets which has been used for training. Hence, we can assume that the datasets have been taken from the online source.

**2.1.4 Used Cars Price Prediction using Supervised Learning Techniques**

**[ISSN: 2249 – 8958, Volume-9 Issue-1S3, December 2019]**

Upon study on this paper they stated that “Multiple and Lasso Regressions are better at predicting price”. Hence Multiple Regression and Lasso Regression Algorithms are been used for implementation of model. The Features that have been used to predict the price of a car are Milage, Brand Name, Model, Body Type, Cylinder, Fuel Capacity, Doors, Leather. . There is no information about the Datasets which has been used for training. Hence, we can assume that the datasets have been taken from the online source.

**2.1.5 Used Cars Price Prediction using Supervised Learning Techniques**

**[IRJET Research paper, Volume-8, April 2021, p-ISSN: 2395-0072]**

Upon study on this paper, it has been found that the Machine Learning Algorithm which has been used for implementation of model is Linear Regression, Ridge Regression, Lasso Regression. There is no information about the Datasets which has been used for training. Hence, we can assume that the datasets have been taken from the online source. A UI which is user friendly and takes input from the user and predicts the price.

**2.2 Disadvantages of Existing System**

By doing the research, It has been found that the models are been trained on the data sets which are available through online resources and these datasets can be far old that it may not match up with the current prices of cars. So when we try to predict the price of a used car the user may get the price which may not be the actual price of this year.

These papers didn’t mention their actually successful accuracy of prediction, which is another issue.

In some papers there is no actual user interface where user can interact with the model and get the price.

**CHAPTER 3**

**PROPOSED SYSTEM**

**3.1 Proposed System**

**3.3 System Requirements**

**3.3.1 Software Requirements**

1. Python 3
2. PyQt5
3. Google Drive API
4. PIP Packages
5. Microsoft C++ Visual Studio
6. PyCharm
7. Auto Py to EXE Generator tool

**3.3.2 Hardware Requirements:**

1. CPU- Intel Dual Core
2. RAM – 2GB
3. OS – Windows 7,8,10,11, Linux
4. Graphic Card – 128MB

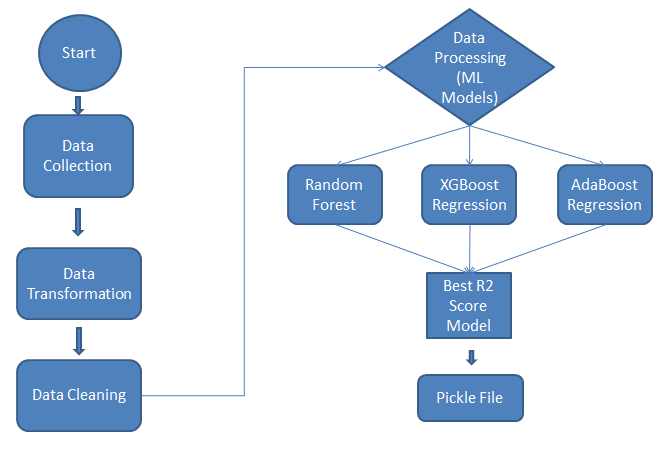
**3.3.3 Implementation Technologies**

* Front End: - PyQt5
* Back End – Python 3
* Cloud – Google Drive
* IDE – Python Virtual Environment
* Packages Imported : requests, json, os, google\_auth\_outhlib, googleapiclient, threading, parese\_url, six,…

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 Proposed System Architecture**



**4.2 Modules**

**4.2.1 GOOGLE DRIVE API:**

In The Google Drive API allows you to create apps that leverage Google Drive cloud storage. You can develop applications that integrate with Google Drive, and create robust functionality in your application using Google Drive API.

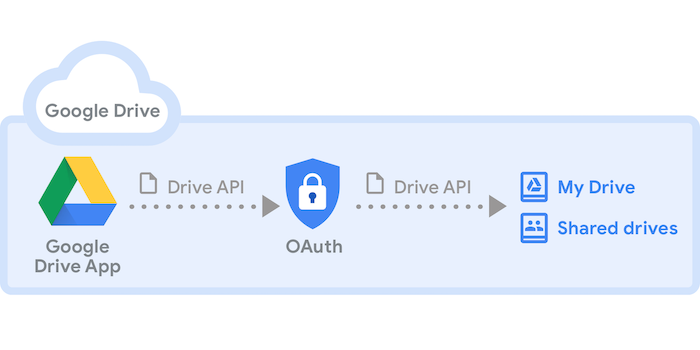
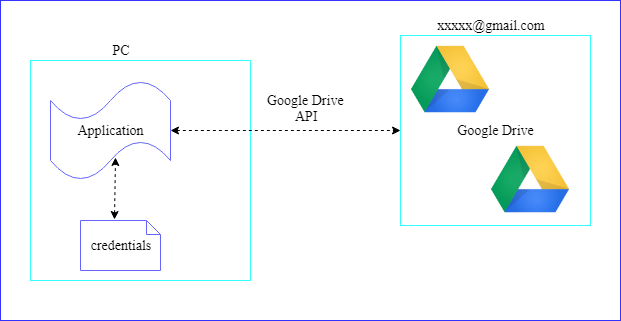


Fig 4.2.1.1

We can use Google Drive API to:

1. Download files from Google Drive and Upload files to Google Drive.
2. [Search for files and folders](https://developers.google.com/drive/api/v3/search-files) stored in Google Drive. Create complex search queries that return any of the file metadata fields in the [Files](https://developers.google.com/drive/api/v3/reference/files) resource.
3. Let users [share files, folders and drives](https://developers.google.com/drive/api/v3/manage-sharing) to collaborate on content.
4. Combine with the [Google Picker API](https://developers.google.com/picker/docs/) to search all files in Google Drive, then return the file name, URL, last modified date, and user.
5. [Create third-party shortcuts](https://developers.google.com/drive/api/v3/integrate-create#create_a_shortcut_to_a_file) that are external links to data stored outside of Drive, in a different data store or cloud storage system.
6. Create a dedicated Drive folder to store your application’s data so that the app cannot access all the user's content stored in Google Drive. See [Store application-specific data](https://developers.google.com/drive/api/v3/appdata).
7. Integrate with the Google Drive UI, which is Google's standard web UI you can use to interact with Drive files. To learn all that you can do with a Drive app that you integrate with the Google Drive UI, see [Drive UI integration overview](https://developers.google.com/drive/api/v3/about-apps)

**4.2.2 Outh 2.0:**



# Fig 4.2.2.1

OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. This specification and its extensions are being developed within the IETF OAuth Working Group.

Google APIs use the [OAuth 2.0 protocol](https://tools.ietf.org/html/rfc6749) for authentication and authorization. Google supports common OAuth 2.0 scenarios such as those for web server, client-side, installed, and limited-input device applications.To begin, obtain OAuth 2.0 client credentials from the [Google API Console](https://console.developers.google.com/). Then your client application requests an access token from the Google Authorization Server, extracts a token from the response, and sends the token to the Google API that you want to access. For an interactive demonstration of using OAuth 2.0 with Google (including the option to use your own client credentials), experiment with the [OAuth 2.0 Playground](https://developers.google.com/oauthplayground/). This page gives an overview of the OAuth 2.0 authorization scenarios that Google supports, and provides links to more detailed content. For details about using OAuth 2.0 for authentication, see [OpenID Connect](https://developers.google.com/identity/protocols/oauth2/openid-connect).

**4.2.3 Google - auth:**

google-auth is the Google authentication library for Python. This library provides the ability to authenticate to Google APIs using various methods. It also provides integration with several HTTP libraries.

Support for Google [Application Default Credentials](https://google-auth.readthedocs.io/en/master/reference/google.auth.html#google.auth.default).

Support for signing and verifying [JWTs](https://google-auth.readthedocs.io/en/master/reference/google.auth.jwt.html#module-google.auth.jwt).

Support for creating [Google ID Tokens](https://google-auth.readthedocs.io/en/master/user-guide.html#identity-tokens).

Support for verifying and decoding [ID Tokens](https://google-auth.readthedocs.io/en/master/reference/google.oauth2.id_token.html#module-google.oauth2.id_token).

Support for Google [Service Account credentials](https://google-auth.readthedocs.io/en/master/reference/google.oauth2.service_account.html#module-google.oauth2.service_account).

Support for Google [Impersonated Credentials](https://google-auth.readthedocs.io/en/master/reference/google.auth.impersonated_credentials.html#module-google.auth.impersonated_credentials).

Support for [Google Compute Engine credentials](https://google-auth.readthedocs.io/en/master/reference/google.auth.compute_engine.html#module-google.auth.compute_engine).

Support for [Google App Engine standard credentials](https://google-auth.readthedocs.io/en/master/reference/google.auth.app_engine.html#module-google.auth.app_engine).

Support for [Identity Pool credentials](https://google-auth.readthedocs.io/en/master/reference/google.auth.identity_pool.html#module-google.auth.identity_pool).

Support for [AWS credentials](https://google-auth.readthedocs.io/en/master/reference/google.auth.aws.html#module-google.auth.aws).

Support for various transports, including [Requests](https://google-auth.readthedocs.io/en/master/reference/google.auth.transport.requests.html#module-google.auth.transport.requests), [urllib3](https://google-auth.readthedocs.io/en/master/reference/google.auth.transport.urllib3.html#module-google.auth.transport.urllib3), and [gRPC](https://google-auth.readthedocs.io/en/master/reference/google.auth.transport.grpc.html" \l "module-google.auth.transport.grpc" \o "google.auth.transport.grpc).

**4.2.4 PyQt5**

PyQt is a GUI widgets toolkit. It is a Python interface for Qt, one of the most powerful, and popular cross-platform GUI library. PyQt was developed by RiverBank Computing Ltd. The latest version of PyQt can be downloaded from its official website − [riverbankcomputing.com](https://riverbankcomputing.com/)

PyQt API is a set of modules containing a large number of classes and functions. While QtCore module contains non-GUI functionality for working with file and directory etc., QtGui module contains all the graphical controls. In addition, there are modules for working with XML (QtXml), SVG (QtSvg), and SQL (QtSql), etc.

PyQt5 Designer

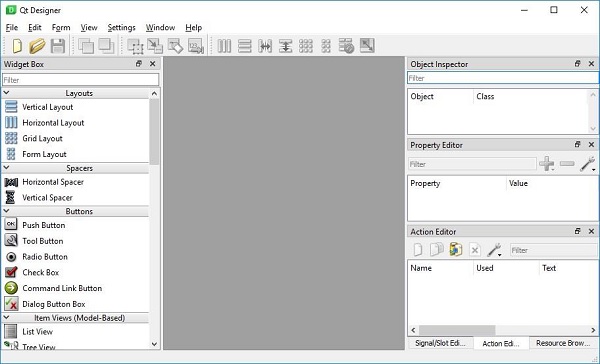


Fig 4.2.4.1

**4.2.5 Auto Py to Exe**

A .py to .exe converter using a simple graphical interface and [PyInstaller](https://www.pyinstaller.org/) in Python.

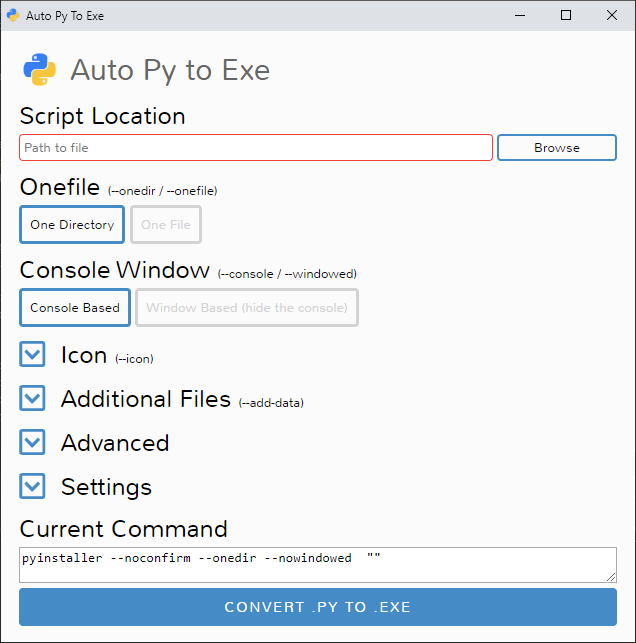


Fig 4.2.5.1

**CHAPTER 5**

**IMPLEMENTATION**

# 5.1 Algorithm:

**5.1.1 Random Forest Regression**

Random forest is yet another powerful and most used **supervised learning**algorithm. It allows quick identification of significant information from**vast datasets.**The biggest advantage of Random forest is that it relies on **collecting various decision trees** to arrive at any solution.

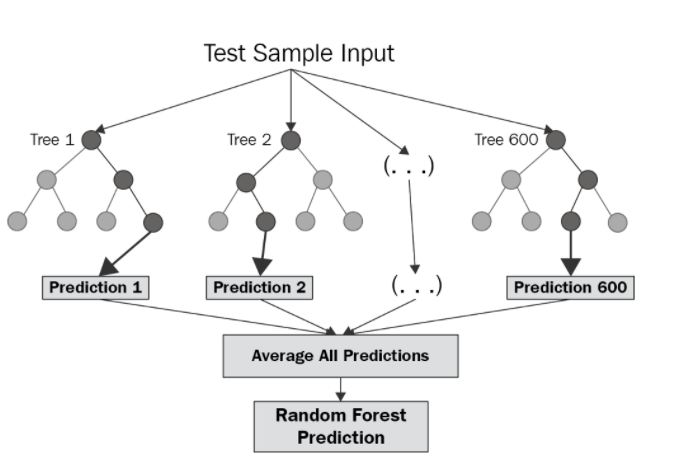
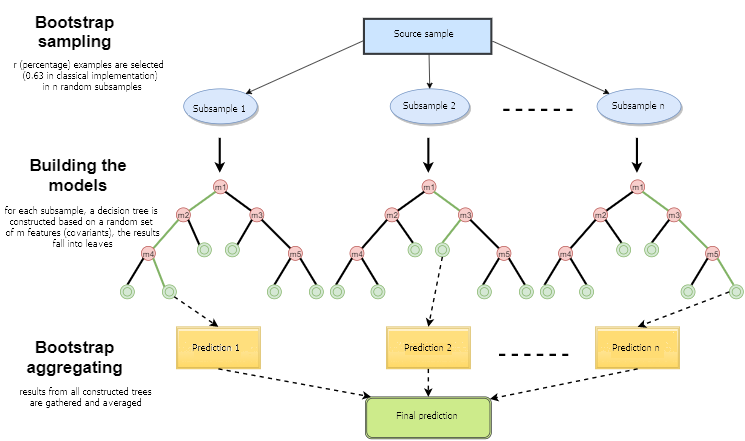


Fig 5.1.1

#### How does Random Forest Works?

Assume “*m*” features in our dataset:

1. Randomly chose “*k”* features satisfying condition *k* < *m.*
2. Among the *k*features, calculate the root node by choosing a node with *the highest Information gain.*
3. Split the node into child nodes.
4. Repeat the previous steps *n* times.
5. You end up with a forest constituting *n* trees.
6. Perform *Bootstrapping,*i.e., combining the results of all Decision Trees.



**5.1.2 AdaBoost Algorithm**

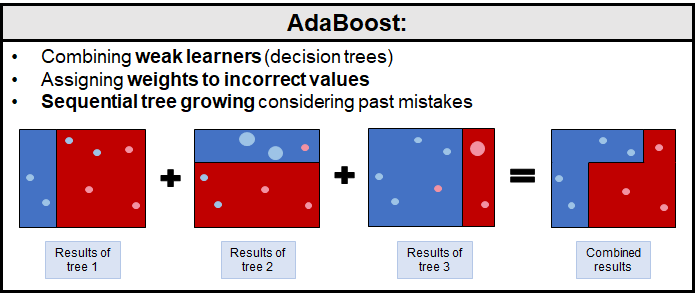
The AdaBoost algorithm is part of the family of boosting algorithms and was first introduced by [Freund & Schapire in 1996](https://pdfs.semanticscholar.org/5fb5/f7b545a5320f2a50b30af599a9d9a92a8216.pdf). It is sequentially growing decision trees as weak learners and punishing incorrectly predicted samples by assigning a larger weight to them after each round of prediction. This way, the algorithm is learning from previous mistakes. The final prediction is the weighted majority vote (or weighted median in case of regression problems).

The **pseudo code** of the AdaBoost algorithm for a classification problem is shown below adapted from [Freund & Schapire in 1996](https://pdfs.semanticscholar.org/5fb5/f7b545a5320f2a50b30af599a9d9a92a8216.pdf) (for regression problems, please refer to the underlying paper):

For *t* in *T* rounds:

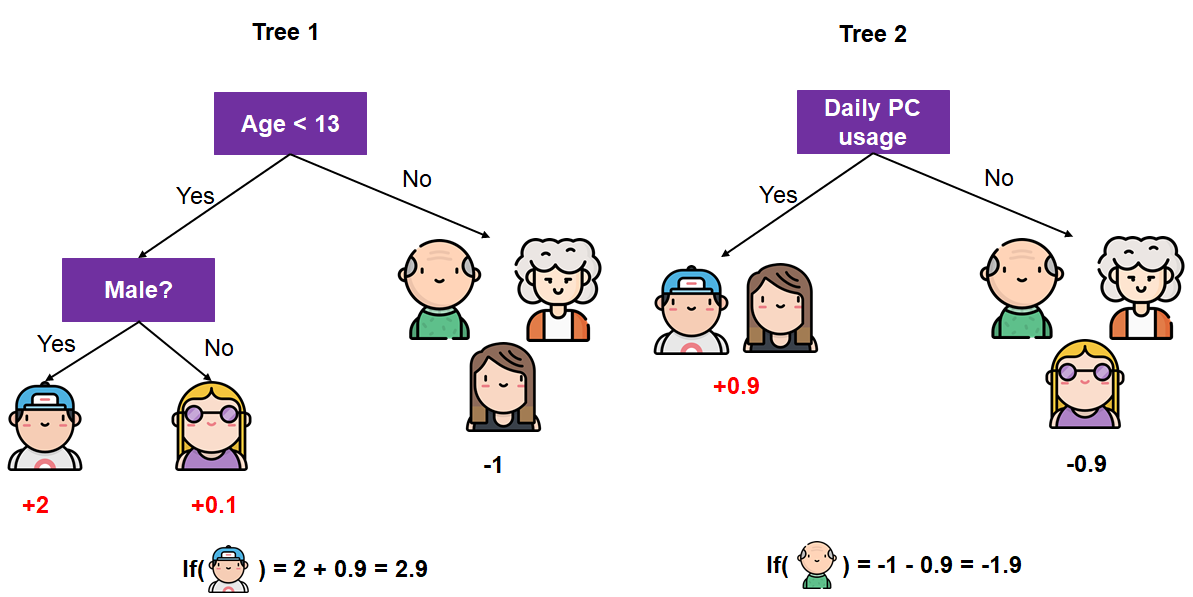
1. Calculate distribution *p* by normalizing the weight vector *w (*the initial weights in *w*for the first round are *1/N*, where *N* represents the number of labeled examples)
2. Grow a weak learner (decision tree) using the distribution *p*; return hypothesis *h*with prediction values for each example
3. Calculate error term *ε* of *h*
4. Assign *β*with*ε/(1-* *ε)*
5. Update the weight vector to *w = w\*β*so that predictions with poor performance will have higher a weight and predictions with better performance will have a lower weight

Output: final hypothesis is the result of a weighted majority vote of all *T* weak learners



**5.1.3 XGBoost**

XGBoost (e**X**treme **G**radient **Boost**ing) is a relatively new algorithm that was introduced by [Chen & Guestrin in 2016](https://dl.acm.org/doi/abs/10.1145/2939672.2939785) and is utilizing the concept of gradient tree boosting. XGBoost was developed to increase speed and performance, while introducing regularization parameters to reduce overfitting. Gradient boosted trees use regression trees (or [CART](https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/)) in a sequential learning process as weak learners. These regression trees are similar to decision trees, however, they use a continuous score assigned to each leaf (i.e. the last node once the tree has finished growing) which is summed up and provides the final prediction. For each iteration*i* which grows a tree *t*, scores *w* are calculated which predict a certain outcome *y*. The learning process aims to [minimize the overall score](https://medium.com/greyatom/a-quick-guide-to-boosting-in-ml-acf7c1585cb5) which is composed of the loss function at *i-1*and the new tree structure of *t*. This allows the algorithm to sequentially grow the trees and learn from previous iterations. Gradient descent is then used to compute the optimal values for each leaf and the overall score of tree *t*. The score is also called the impurity of the predictions of a tree.



The loss function in the above algorithm contains a regularization or penalty term Ω whose goal it is to reduce the complexity of the regression tree functions. This parameter can be tuned and can take values equal or greater than 0. If it is set to 0, then there is no difference between the prediction results of gradient boosted trees and XGBoost. In addition, Chen & Guestrin introduce shrinkage (i.e. a learning rate) and column subsampling (randomly selecting a subset of features) to this gradient tree boosting algorithm which allows further reduction of overfitting. It therefore adds the methods to handle overfitting introduced in AdaBoost (the learning rate) and random forests (column or feature subsampling) to the regularization parameter found in stochastic gradient descent models.

**5.2 Implementation steps :**

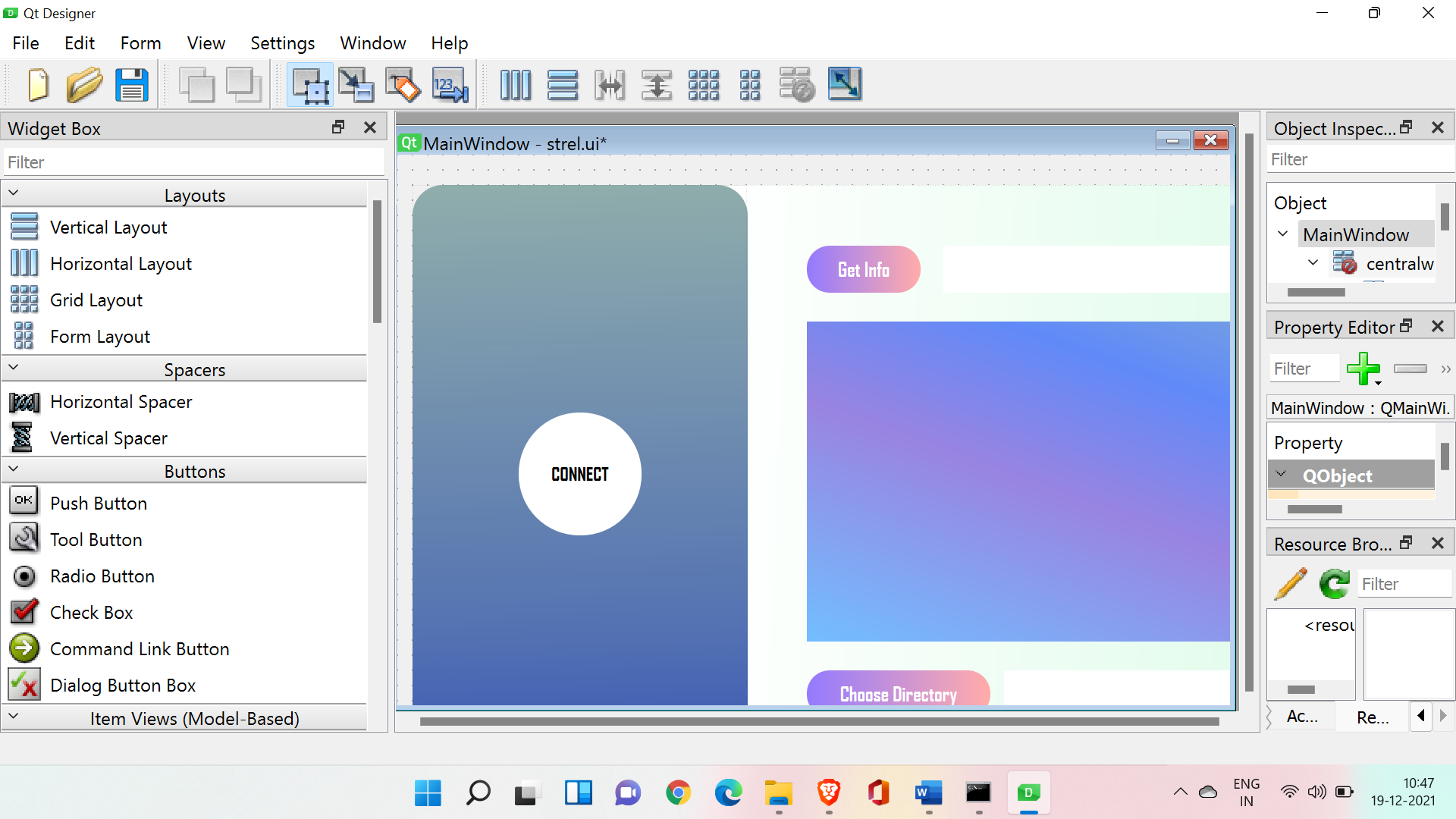


Fig 5.2.1

Designing the front end for the project using PyQt5Designer tool. It produces the output in the form of .ui file which is a XML file. The XML file is converted to .py i.e. Python file in order add functionalities for the application

After adding the functionalities to the code the code is again converted to .exe file i.e. a runnable application with the help of Auto .py to .exe generator tool.

**5.3 Source Code :**

**Strel.py**

from \_\_future\_\_ import print\_function

from threading import \*

from PyQt5.QtCore import \*

from PyQt5 import QtCore

from PyQt5 import QtGui

from google\_auth\_oauthlib.flow import Flow, InstalledAppFlow

from googleapiclient.discovery import build

from googleapiclient.http import MediaFileUpload, MediaIoBaseDownload

from google.auth.transport.requests import Request

from PyQt5.QtWidgets import QMessageBox

from PyQt5 import QtWidgets

from parse\_url import parse\_url

from six.moves import urllib\_parse

#from Google import Create\_Service

from PyQt5.QtWidgets import QDialog,QMainWindow, QApplication, QFileDialog

from PyQt5.uic import loadUi

import pickle

import sys

import glob

import json

import os

import os.path as osp

import re

import shutil

import sys

import tempfile

import textwrap

import time

import warnings

import requests

import six

import tqdm

global progess\_total

global curr\_progress

progess\_total=0

curr\_progress=0

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(887, 543)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.widget = QtWidgets.QWidget(self.centralwidget)

self.widget.setGeometry(QtCore.QRect(10, 20, 221, 501))

self.widget.setStyleSheet("background-color: qlineargradient(spread:pad, x1:0.506, y1:0, x2:0.517, y2:1, stop:0 rgba(143, 173, 171, 255), stop:0.585227 rgba(84, 114, 178, 255), stop:1 rgba(24, 53, 184, 255));\n"

"border-radius: 20px;")

self.widget.setObjectName("widget")

self.connect = QtWidgets.QPushButton(self.widget)

self.connect.setGeometry(QtCore.QRect(70, 150, 81, 81))

self.connect.setStyleSheet("background-color: rgb(255, 255, 255);\n"

"font: 75 9pt \"Agency FB\";\n"

"font-weight: Bold;\n"

"border-radius: 40px;\n"

"")

self.connect.setObjectName("connect")

self.exit = QtWidgets.QPushButton(self.widget)

self.exit.setGeometry(QtCore.QRect(70, 440, 75, 23))

self.exit.setStyleSheet("background-color: rgb(167, 58, 7);\n"

"color: rgb(255, 255, 255);\n"

"border-radius: 10px;")

self.exit.setObjectName("exit")

self.rezero = QtWidgets.QWidget(self.centralwidget)

self.rezero.setGeometry(QtCore.QRect(200, 20, 671, 501))

self.rezero.setStyleSheet("background-color: qlineargradient(spread:pad, x1:0, y1:0.489, x2:1, y2:0.517, stop:0 rgba(255, 255, 255, 255), stop:0.977273 rgba(204, 250, 221, 255));\n"

"border-radius: 20px;")

self.rezero.setObjectName("rezero")

self.inputid = QtWidgets.QLineEdit(self.rezero)

self.inputid.setGeometry(QtCore.QRect(160, 40, 481, 31))

self.inputid.setStyleSheet("background-color: rgb(255, 255, 255);\n"

"font: 8pt \"Papyrus\";")

self.inputid.setObjectName("inputid")

self.logo = QtWidgets.QTextEdit(self.rezero)

self.logo.setGeometry(QtCore.QRect(70, 360, 571, 91))

self.logo.setStyleSheet("background-color: qlineargradient(spread:pad, x1:0, y1:1, x2:1, y2:0, stop:0 rgba(255, 114, 246, 255), stop:0.375 rgba(153, 132, 225, 255), stop:0.613636 rgba(96, 138, 248, 255), stop:1 rgba(245, 147, 201, 255));\n"

"color: rgb(255, 255, 255);\n"

"font: 10pt \"Centaur\";\n"

"font-weight: bold;\n"

"border-radius: 0px;")

self.logo.setObjectName("logo")

self.getinfo = QtWidgets.QPushButton(self.rezero)

self.getinfo.setGeometry(QtCore.QRect(70, 40, 75, 31))

self.getinfo.setStyleSheet("background-color: qlineargradient(spread:pad, x1:0, y1:0.489, x2:1, y2:0.489, stop:0 rgba(147, 122, 255, 255), stop:0.982955 rgba(255, 171, 171, 255));\n"

"color: rgb(255, 255, 255);\n"

"border-radius: 15px;\n"

"font: 75 10pt \"Agency FB\";\n"

"font-weight: Bold;\n"

"")

self.getinfo.setObjectName("getinfo")

self.dirchooser = QtWidgets.QPushButton(self.rezero)

self.dirchooser.setGeometry(QtCore.QRect(70, 320, 121, 31))

self.dirchooser.setStyleSheet("background-color: qlineargradient(spread:pad, x1:0, y1:0.489, x2:1, y2:0.489, stop:0 rgba(147, 122, 255, 255), stop:0.982955 rgba(255, 171, 171, 255));\n"

"color: rgb(255, 255, 255);\n"

"border-radius: 15px;\n"

"font: 75 10pt \"Agency FB\";\n"

"font-weight: Bold;\n"

"")

self.dirchooser.setObjectName("dirchooser")

self.download = QtWidgets.QPushButton(self.rezero)

self.download.setGeometry(QtCore.QRect(550, 320, 91, 31))

self.download.setStyleSheet("background-color: qlineargradient(spread:pad, x1:0, y1:0.489, x2:1, y2:0.489, stop:0 rgba(147, 122, 255, 255), stop:0.982955 rgba(255, 171, 171, 255));\n"

"color: rgb(255, 255, 255);\n"

"border-radius: 15px;\n"

"font: 75 10pt \"Agency FB\";\n"

"font-weight: Bold;\n"

"")

self.download.setObjectName("download")

self.listid = QtWidgets.QTableWidget(self.rezero)

self.listid.setEnabled(True)

self.listid.setGeometry(QtCore.QRect(70, 90, 571, 211))

self.listid.setStyleSheet("font: 10pt \"Centaur\";\n"

"font-weight: bold;\n"

"background-color: qlineargradient(spread:pad, x1:0, y1:1, x2:1, y2:0, stop:0 rgba(114, 190, 255, 255), stop:0.375 rgba(153, 132, 225, 255), stop:0.613636 rgba(96, 138, 248, 255), stop:1 rgba(148, 192, 194, 255));\n"

"color: rgb(255, 255, 255);\n"

"font-weight: bold;\n"

"border-radius: 0px;")

self.listid.setFrameShape(QtWidgets.QFrame.NoFrame)

self.listid.setFrameShadow(QtWidgets.QFrame.Plain)

self.listid.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustIgnored)

self.listid.setProperty("showDropIndicator", False)

self.listid.setSelectionMode(QtWidgets.QAbstractItemView.NoSelection)

self.listid.setShowGrid(False)

self.listid.setGridStyle(QtCore.Qt.NoPen)

self.listid.setWordWrap(False)

self.listid.setObjectName("listid")

self.listid.setColumnCount(5)

self.listid.setRowCount(0)

item = QtWidgets.QTableWidgetItem()

self.listid.setHorizontalHeaderItem(0, item)

item = QtWidgets.QTableWidgetItem()

self.listid.setHorizontalHeaderItem(1, item)

item = QtWidgets.QTableWidgetItem()

self.listid.setHorizontalHeaderItem(2, item)

item = QtWidgets.QTableWidgetItem()

self.listid.setHorizontalHeaderItem(3, item)

item = QtWidgets.QTableWidgetItem()

self.listid.setHorizontalHeaderItem(4, item)

self.listid.horizontalHeader().setVisible(False)

self.listid.horizontalHeader().setHighlightSections(False)

self.listid.verticalHeader().setVisible(False)

self.listid.verticalHeader().setHighlightSections(False)

self.progress = QtWidgets.QProgressBar(self.rezero)

self.progress.setGeometry(QtCore.QRect(70, 460, 571, 23))

self.progress.setAutoFillBackground(False)

self.progress.setProperty("value", 0)

self.progress.setTextVisible(False)

self.progress.setObjectName("progress")

self.dirlabel = QtWidgets.QLineEdit(self.rezero)

self.dirlabel.setEnabled(True)

self.dirlabel.setGeometry(QtCore.QRect(200, 320, 331, 31))

self.dirlabel.setStyleSheet("background-color: rgb(255, 255, 255);\n"

"font: 8pt \"Papyrus\";\n"

"")

self.dirlabel.setObjectName("dirlabel")

self.rezero.raise\_()

self.widget.raise\_()

MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

self.connect.setText(\_translate("MainWindow", "CONNECT"))

self.exit.setText(\_translate("MainWindow", "EXIT"))

self.getinfo.setText(\_translate("MainWindow", "Get Info"))

self.dirchooser.setText(\_translate("MainWindow", "Choose Directory"))

self.download.setText(\_translate("MainWindow", "Download"))

self.listid.setSortingEnabled(False)

item = self.listid.horizontalHeaderItem(0)

item.setText(\_translate("MainWindow", "blank space"))

item = self.listid.horizontalHeaderItem(1)

item.setText(\_translate("MainWindow", "File Name"))

item = self.listid.horizontalHeaderItem(2)

item.setText(\_translate("MainWindow", "Type"))

item = self.listid.horizontalHeaderItem(3)

item.setText(\_translate("MainWindow", "Size"))

item = self.listid.horizontalHeaderItem(4)

item.setText(\_translate("MainWindow", "Progress"))

class WorkerThread(QThread):

update\_progress=pyqtSignal(list)

terminal\_progress=pyqtSignal(str)

msg\_progress=pyqtSignal(bool)

def run(self):

use\_cookies=True

verify=True

proxy=None

speed=None

quiet=False

resume=False

global set\_id

global fname

dc=0

zerotwo=False

for i in set\_id:

dc+=1

url="https://drive.google.com/uc?id="+i

if fname[-1]!="/":

fname=fname+"/"

fff=fname+set\_id[i][0]

fff=fff.replace("/","\\")

if not os.path.exists(fff):

os.makedirs(fff)

output=fff+set\_id[i][1]

CHUNK\_SIZE = 512 \* 1024

home = osp.expanduser("~")

url\_origin = url

sess = requests.session()

cache\_dir = osp.join(home, ".cache", "gdown")

if not osp.exists(cache\_dir):

os.makedirs(cache\_dir)

cookies\_file = osp.join(cache\_dir, "cookies.json")

if osp.exists(cookies\_file) and use\_cookies:

with open(cookies\_file) as f:

cookies = json.load(f)

for k, v in cookies:

sess.cookies[k] = v

if proxy is not None:

sess.proxies = {"http": proxy, "https": proxy}

print("Using proxy:", proxy, file=sys.stderr)

parsed = urllib\_parse.urlparse(url)

query = urllib\_parse.parse\_qs(parsed.query)

is\_gdrive = parsed.hostname in ["drive.google.com", "docs.google.com"]

is\_download\_link = parsed.path.endswith("/uc")

file\_id = None

if "id" in query:

file\_ids = query["id"]

if len(file\_ids) == 1:

file\_id = file\_ids[0]

else:

patterns = [r"^/file/d/(.\*?)/view$", r"^/presentation/d/(.\*?)/edit$"]

for pattern in patterns:

match = re.match(pattern, parsed.path)

if match:

file\_id = match.groups()[0]

break

gdrive\_file\_id, is\_gdrive\_download\_link = file\_id, is\_download\_link

url = "https://drive.google.com/uc?id={id}".format(id=gdrive\_file\_id)

url\_origin = url

is\_gdrive\_download\_link = True

headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36"}

while True:

try:

res = sess.get(url, headers=headers, stream=True, verify=verify)

except requests.exceptions.ProxyError as e:

print("An error has occurred using proxy:", proxy, file=sys.stderr)

print(e, file=sys.stderr)

return

with open(cookies\_file, "w") as f:

cookies = [

(k, v)

for k, v in sess.cookies.items()

if not k.startswith("download\_warning\_")

]

json.dump(cookies, f, indent=2)

if "Content-Disposition" in res.headers:

break

if not (gdrive\_file\_id and is\_gdrive\_download\_link):

break

try:

url = ""

contents=res.text

for line in contents.splitlines():

m = re.search(r'href="(\/uc\?export=download[^"]+)', line)

if m:

url = "https://docs.google.com" + m.groups()[0]

url = url.replace("&amp;", "&")

break

m = re.search("confirm=([^;&]+)", line)

if m:

confirm = m.groups()[0]

url = re.sub(r"confirm=([^;&]+)", r"confirm={}".format(confirm), url)

break

m = re.search('"downloadUrl":"([^"]+)', line)

if m:

url = m.groups()[0]

url = url.replace("\\u003d", "=")

url = url.replace("\\u0026", "&")

break

m = re.search('<p class="uc-error-subcaption">(.\*)</p>', line)

except:

print("Access denied with the following error:")

#print(contents)

return

p\_count=0

if gdrive\_file\_id and is\_gdrive\_download\_link:

m = re.search('filename="(.\*)"', res.headers["Content-Disposition"])

filename\_from\_url = m.groups()[0]

else:

filename\_from\_url = osp.basename(url)

if output is None:

output = filename\_from\_url

output\_is\_path = isinstance(output, six.string\_types)

if output\_is\_path and output.endswith(osp.sep):

if not osp.exists(output):

os.makedirs(output)

output = osp.join(output, filename\_from\_url)

if output\_is\_path:

existing\_tmp\_files = glob.glob("{}\*".format(output))

if resume and existing\_tmp\_files:

tmp\_file = existing\_tmp\_files[0]

else:

resume = False

tmp\_file = tempfile.mktemp(suffix=tempfile.template,prefix=osp.basename(output),dir=osp.dirname(output),)

f = open(tmp\_file, "ab")

else:

tmp\_file = None

f = output

if tmp\_file is not None and f.tell() != 0:

headers["Range"] = "bytes={}-".format(f.tell())

res = sess.get(url, headers=headers, stream=True, verify=verify)

# if resume:

# print("Resume:", tmp\_file, file=sys.stderr)

# print("From:", url\_origin, file=sys.stderr)

# print("To:",osp.abspath(output) if output\_is\_path else output,file=sys.stderr,)

try:

total = res.headers.get("Content-Length")

if total is not None:

total = int(total)

t\_start = time.time()

for chunk in res.iter\_content(chunk\_size=CHUNK\_SIZE):

f.write(chunk)

p\_count=p\_count+len(chunk)

percent=int((p\_count/total)\*100)

self.update\_progress.emit([set\_id[i][2],percent,p\_count])

if speed is not None:

elapsed\_time\_expected = 1.0 \* pbar.n / speed

elapsed\_time = time.time() - t\_start

if elapsed\_time < elapsed\_time\_expected:

time.sleep(elapsed\_time\_expected - elapsed\_time)

if tmp\_file:

f.close()

shutil.move(tmp\_file, output)

except IOError as e:

print(e, file=sys.stderr)

return

finally:

sess.close()

if dc==len(set\_id):

zerotwo=True

self.terminal\_progress.emit("{yo} has been downloaded".format(yo=set\_id[i][1][1:]))

self.msg\_progress.emit(zerotwo)

global service

global set\_id

global fname

global c

c=1

set\_id={}

class Res(QMainWindow, Ui\_MainWindow):

def \_\_init\_\_(self,parent = None):

global progess\_total,curr\_progress

super(Res,self).\_\_init\_\_(parent)

self.setupUi(self)

self.rect=QRect(200,20,0,501)

self.rezero.setGeometry(self.rect)

self.download.setEnabled(False)

self.clicked=False

"""item1=self.listid.horizontalHeaderItem(0)

item1.setForeground(QtGui.QColor(255, 0, 0))

item1.setBackground(QtGui.QColor(255, 0, 0))

self.listid.setHorizontalHeaderItem(0,item1)"""

self.listid.setColumnWidth(0,2)

self.listid.setColumnWidth(1,240)

self.listid.setColumnWidth(2,65)

self.listid.setColumnWidth(3,100)

self.listid.setColumnWidth(4,90)

self.listid.setRowCount(1)

self.listid.setItem(0,1,QtWidgets.QTableWidgetItem("File Name"))

self.listid.setItem(0,2,QtWidgets.QTableWidgetItem("Type"))

self.listid.setItem(0,3,QtWidgets.QTableWidgetItem("Size"))

self.listid.setItem(0,4,QtWidgets.QTableWidgetItem("Progress"))

#self.listid.setColumnWidth(4,60)

self.logo.setReadOnly(True)

self.dirlabel.setReadOnly(True)

self.getinfo.setEnabled(False)

#self.dirlabel.setReadOnly(True)

self.dirchooser.clicked.connect(self.choose\_folder)

self.connect.clicked.connect(self.connect\_drive)

self.exit.clicked.connect(self.exit\_st)

self.connect.setStyleSheet('''

QPushButton

{

background-color: rgb(255, 255, 255);

font: 75 9pt "Agency FB";

font-weight: Bold;

border-radius: 40px;

}

QPushButton::hover

{

color: white;

font: 75 10pt "Agency FB";

font-weight: Bold;

background-color: qlineargradient(spread:pad, x1:0, y1:0.489, x2:1, y2:0.489, stop:0 rgba(147, 122, 255, 255), stop:0.982955 rgba(255, 171, 171, 255));

}

''')

self.getinfo.clicked.connect(self.drive\_info)

self.download.clicked.connect(self.file\_download)

def exit\_st(self):

exit()

def file\_download(self):

self.download.setEnabled(False)

self.dirchooser.setEnabled(False)

self.worker=WorkerThread()

self.worker.start()

self.worker.update\_progress.connect(self.update\_progress\_percent)

self.worker.terminal\_progress.connect(self.update\_terminal)

self.worker.msg\_progress.connect(self.msg\_box)

def update\_terminal(self,yoyo):

self.logo.append(" "+yoyo)

def msg\_box(self,val):

if val==True:

res=QMessageBox.information(self, "SSup", "Download Completed",QMessageBox.Ok)

if res==QMessageBox.Ok:

exit()

def update\_progress\_percent(self,val):

global curr\_progress,progess\_total

curr\_progress=curr\_progress+val[2]

self.progress.setValue(int((curr\_progress/progess\_total)\*100))

self.listid.setItem(val[0],4,QtWidgets.QTableWidgetItem(str(val[1])+"%"))

if int((curr\_progress/progess\_total)\*100)>=100:

self.progress.setValue(100)

def choose\_folder(self):

global fname

fname=str(QFileDialog.getExistingDirectory(self, 'Select Directory'))

self.dirlabel.setText(fname)

self.download.setEnabled(True)

def connect\_drive(self):

global service

self.connect.setEnabled(False)

c\_file="token.json"

api\_name="drive"

api\_version="v3"

scope=["https://www.googleapis.com/auth/drive"]

service=self.Create\_Service(c\_file,api\_name,api\_version,scope)

if service is None:

self.logo.append(" Connection Failed")

self.connect.setEnabled(True)

else:

self.logo.append(" Connection Made Successfull")

self.getinfo.setEnabled(True)

self.anime=QPropertyAnimation(self.rezero,b'geometry')

self.anime.setDuration(1200)

self.anime.setStartValue(QRect(200,20,0,501))

self.anime.setEndValue(QRect(200,20,671,501))

self.anime.start()

def get\_child(self,query):

global service

page\_token=None

result\_set=[]

while True:

response=service.files().list(q=query,fields='nextPageToken,files(id,name,mimeType,parents,size,fileExtension)',pageToken=page\_token).execute()

result\_set.extend(response.get('files'))

page\_token=response.get('nextPageToken',None)

if page\_token is None:

break

return result\_set

def recursive\_pro(self,ff,path):

global set\_id

global c

global service

global progess\_total

if ff['mimeType']=="application/vnd.google-apps.folder":

query=f"'{ff['id']}' in parents and trashed=false"

path=path+"/"+ff['name']

x=self.get\_child(query)

for i in x:

self.recursive\_pro(i,path)

else:

set\_id[ff['id']]=[path,"/"+ff['name'],c]

progess\_total=progess\_total+int(ff['size'])

self.listid.setRowCount(c+1)

self.listid.setItem(c,1,QtWidgets.QTableWidgetItem(ff['name']))

#self.listid.setItem(c,1,QtWidgets.QTableWidgetItem(ff['id']))

self.listid.setItem(c,2,QtWidgets.QTableWidgetItem(ff['fileExtension']))

self.listid.setItem(c,3,QtWidgets.QTableWidgetItem(ff['size']))

self.listid.setItem(c,4,QtWidgets.QTableWidgetItem("0%"))

c+=1

def validate(self,xlink):

if ("drive.google.com" not in xlink) or ("folders" not in xlink):

return xlink,False

url=xlink[xlink.rfind("/")+1:]

if "?" in url:

url=url[:url.rfind("?")]

return url,True

def drive\_info(self):

global service

self.getinfo.setEnabled(False)

xlink=self.inputid.text()

link,valid=self.validate(xlink)

if valid:

query=f"'{link}' in parents and trashed=false"

"""response=service.files().list(q=query).execute()

files=response.get('files')

nextPageToken=response.get('nextPageToken')

while nextPageToken:

response=service.files().list(q=query,pageToken=nextPageToken).execute()

files.extend(response.get('files'))

nextPageToken=response.get('nextPageToken')

for i in files:

print(i)"""

x=self.get\_child(query)

for i in x:

self.recursive\_pro(i,"Shared\_folder")

else:

self.getinfo.setEnabled(True)

self.logo.append(" The URL you have Entered is incorrect!")

self.logo.append(" Please enter URL which is in this format:")

self.logo.append(" https://drive.google.com/drive/u/0/folders/1lTc0-egg2f09I3r3PglxNAj8Ho44Bm")

self.logo.append(" OR")

self.logo.append(" https://drive.google.com/drive/folders/1lTc0-egg2f09I3r3PglxNAj8Ho44Bm?usp=sharing")

def Create\_Service(self,client\_secret\_file, api\_name, api\_version, \*scopes):

CLIENT\_SECRET\_FILE=client\_secret\_file

API\_SERVICE\_NAME=api\_name

API\_VERSION = api\_version

SCOPES = [scope for scope in scopes[0]]

cred = None

pickle\_file = f'token\_{API\_SERVICE\_NAME}\_{API\_VERSION}.pickle'

if os.path.exists(pickle\_file):

with open(pickle\_file, 'rb') as token:

cred=pickle.load(token)

if not cred or not cred.valid:

if cred and cred.expired and cred.refresh\_token:

cred.refresh(Request())

else:

flow = InstalledAppFlow.from\_client\_secrets\_file(CLIENT\_SECRET\_FILE, SCOPES)

cred = flow.run\_local\_server()

with open(pickle\_file,'wb') as token:

pickle.dump(cred, token)

try:

service = build(API\_SERVICE\_NAME, API\_VERSION, credentials=cred)

return service

except Exception as e:

return None

app=QApplication(sys.argv)

mainwindow=Res()

widget=QtWidgets.QStackedWidget()

widget.addWidget(mainwindow)

widget.setWindowFlags(QtCore.Qt.FramelessWindowHint)

widget.setAttribute(QtCore.Qt.WA\_TranslucentBackground)

widget.setFixedWidth(887)

widget.setFixedHeight(542)

widget.show()

app.exec\_()

**CHAPTER 7**

**RESULTS**

The Fig 7.1 shows the basic view of application when it is opened i.e. during startup. It has a connect button and an exit button. Connect button will help you to connect the google account in order to access the other functionalities.

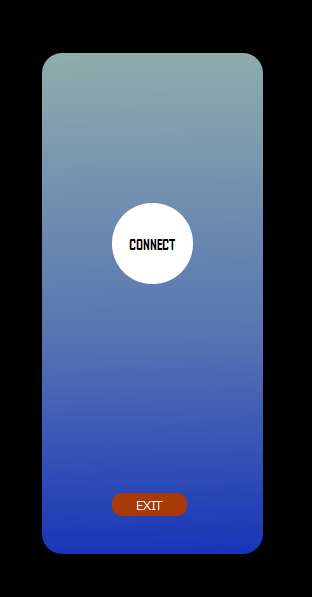


Fig 7.1

After connecting to a google account, a separate panel will slide from the right side on the monitor screen which will have other functionalities as show in the fig 7.2.

# E:\1.png

Fig 7.2

In fig 7.2 we can see it has a text label on left and a button “Get Info” on right. In the text label we can give the URL as an input. After clicking the “Get Info” button the file data will be viewed as shown in the fig 7.3

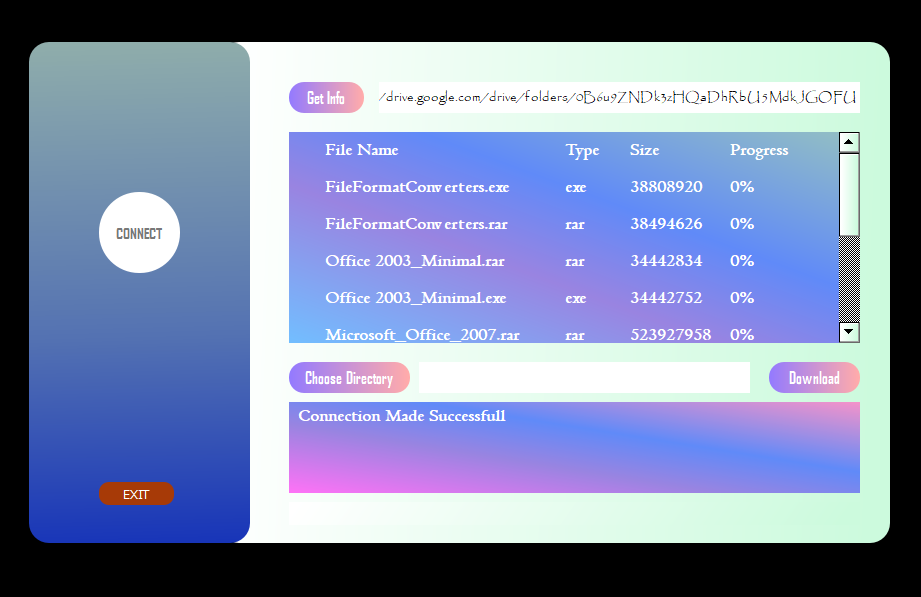


Fig 7.3

After getting the data we need to choose the directory where we need to download and for this there is a choose directory button below the table view on the left side. By clicking it we can choose directory as shown in fig 7.4.

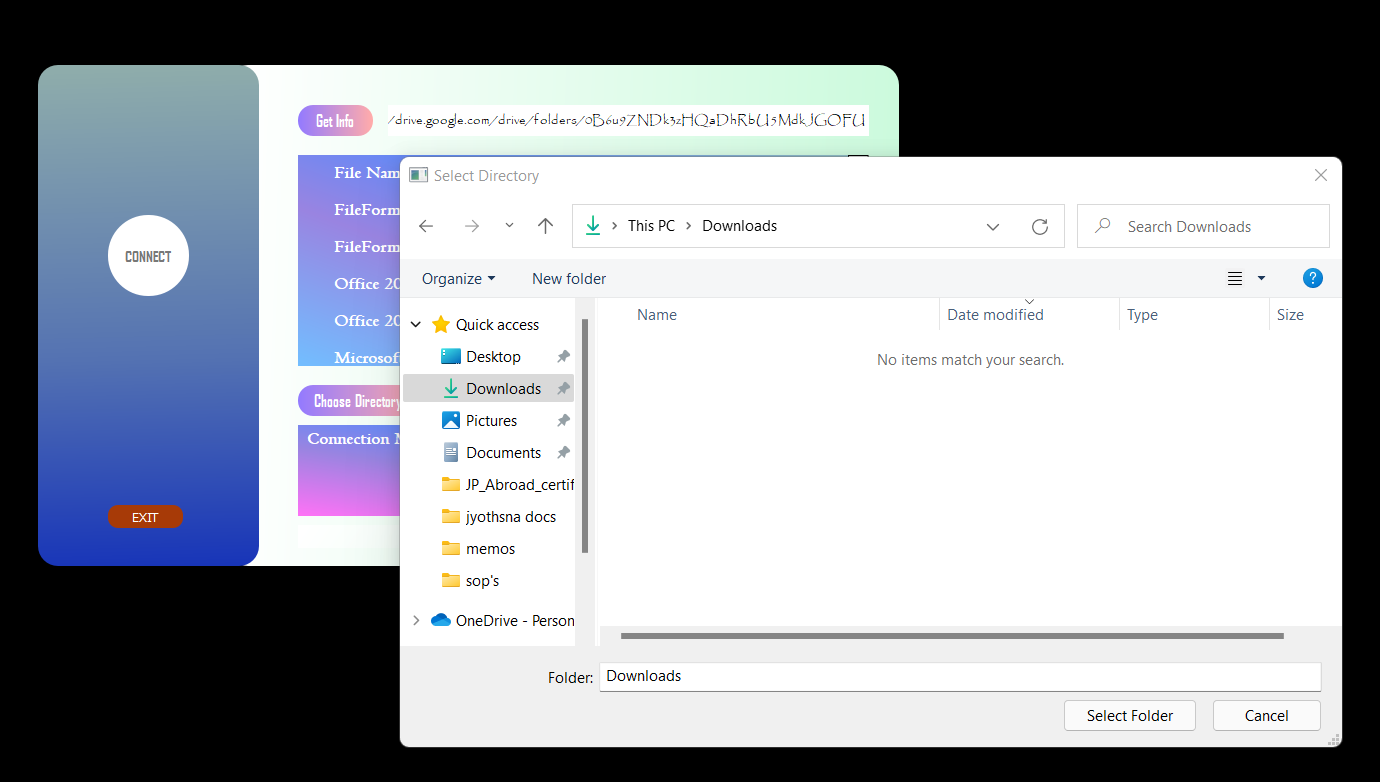


Fig 7.4

click the “Download” button to download the folder. The progress will be updated accordingly besides each file data as well as a progress bar will also be viewed at bottom.



Fig 7.5

**CHAPTER 8**

**CONCLUSION**

The problem which was identified from Google drive that is whenever we try to download a folder from Google Drive using any web browser it goes through some certain process. As the browsers are not able download a directory, the google drive will compress the folder i.e. to convert it into a downloadable zip file. Here the number of zip file is not a unit number. It depends on the size of the folder. Google Drive will produce a max of 2 GB zip files, suppose for an example we have a folder which has contents of total size upto 8 GB, then Google drive will generate 4 files. Google drive may take time to zips the folder and it depends on many factors such as server, file type, bandwidth. It may take from 30 seconds to 3 Hours. After the compressing step is done the files get downloaded and again, we need to extract them in our local file system. The whole thing is a time taking process and during extraction RAM+CPU time is also consumed. The above problem will be solved by our tool by taking input as an URL and download all the files and grouping them in the same directory structure format as viewed in Google Drive. Here the Time is saved that was spent during extracting and zipping by user and the browser.

**FUTURE ENHANCEMENT**

Addition Functionalities will be added such as different cloud supports for example: Mega, icloud, DropBox because this is the same problem with other clouds and also for the next model uploading folder at a time with same directory structure functionality will be added.

# REFERENCES