PROJECT AUDIT REPORT

Assessment Task 3

Submitted by:
Team 5
Ginu T. George
Hari Sankar
Kevin Jojo

Table of Contents

Team Contribution

HARI SANKAR MANGATTU JAGATH PRADEEP - TECHNICAL LEAD

This person translates the business requirements into a technical solution. Because of this responsibility, it is beneficial to have the Technical Lead involved in the planning phase to hear the business requirements from the customer's point of view and ask questions.

The Technical Lead is the development team leader and works with the developers to provide technical details and estimates for the proposed solution. This information is used by the Project Manager to create the Statement of Work and the Work Breakdown Structure documents for the software project.

It is critical that the Technical Lead can effectively communicate the status of the software project to the Project Manager so that issues or variances can be effectively addressed as soon as possible.

Work Contributed:

(1) Python Program for Updating the code's efficiency.

(2) Creating different files each based on a different primary disease (Only one row for a patient).

(3) Preparing the files to be fed as an input to the data mining algorithms

(4) Threshold of the cross-validation can be changed between 0.5 and 1

(5)Audit Report

(6)Cycle Report

KEVIN JOJO - SOFTWARE TESTER

The Software Testers ensure that the software solution meets the business requirements and that it is free of bugs, errors, and defects.

In the test planning and preparation phases of the software testing, Software Testers should review and contribute to test plans, as well as be analyzing, reviewing, and assessing technical requirements and design specifications.

Software Testers are involved in identifying test conditions and creating test designs, test cases, test procedure specifications, and test data, and may automate or help to automate the tests.

Work Contributed:

(1) Python Program for Creating different files each based on gender(Only one row for a patient).

(2) Generalizing data mining algorithm functions

(3) Leave one out cross-validation is to be done on the CSV files

(4) Burndown

(5)Velocity Chart

(6)Audit Report

(7)Cycle Report

Ginu Tholooparampil George - PROJECT MANAGER

The Project Manager is responsible for knowing the “who, what, where, when, and why” of the software project. This means knowing the stakeholders of the project and being able to effectively communicate with each of them.

The Project Manager is also responsible for creating and managing the project budget and schedule as well as processes including scope management, issues management, and risk management.

The Project Manager also oversees software testing, delivery, and formal acceptance by the customer. Then the Project Manager performs a project review with the software development team to document any lessons learned from the software development processes.

Work Contributed:

(1)Python Program for Creating different files each based on insurance (Only one row for a patient)

(2) Generalizing data mining algorithm functions

(3) Github

(4)Trello

(5)Cycle Report

(6)Audit Report

(7)Presentation – PPT

(8)Test Cases

Abstract

In this audit report, all the details regarding the final release of the project are explained. Also, there were client meetings each week and as per the client feedback, changes were made and all the user stories, client acceptance are included in this report.

The report contains the final code for the final release, client acceptance of the final release, and feedback. The picturization of the time taken for alpha and beta release and final is represented using a chart diagram were, it shows both the planned time and the actual time taken to complete the project.

.

Code explanation

The below shown are the imported libraries we used for this project:

```
import pandas as pd
import re
import sys
import time
import numpy  # Slicing individual rows of data.
from numpy import array
from numpy import reshape
# Scikit-Learn data preparation.
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
# Skikit-Learn prediction algorithms.
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF  # For GaussianProcessClassifier.
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
# Didn't use histogram-based gradient boosting.
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier
# Scikit-Learn measures of prediction accuracy.
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

The main library used is pandas we were able to load and manipulate CSV files using the pandas' library. The other important libraries used are sklearn and NumPy. Sklearn is the library that provides the necessary data mining algorithms.

The below-shown codes are for the codes which are used to run different algorithms:

```
def analyse(my_dataset, alg_no, column_Name, data):
    column_names = ["Threshold", "TP", "FP", "FN", "TN", "Precision_P", "Recall_P"]
    final_df = pd.DataFrame(columns=column_names)
    num_positives = 0.001
    no_of_ones = list(my_dataset.EXPIRE_FLAG).count(1)
    if no_of_ones > 0:
        num_positives = list(my_dataset.EXPIRE_FLAG).count(1)
    num_lines = len(my_dataset.index)  # One row of headers and 510 rows of data.
    number_of_column = len(my_dataset.columns)
    column_narray = []
    for n in range(0, number_of_column):
        column_narray.append(n)
    y_list = column_narray.pop()
    X = my_dataset.iloc[:, column_narray].values  # Columns of inputs.
    y = my_dataset.iloc[:, y_list].values  # Last column is output, Yes or No.


    # Store results in a big array.
    all_results = [[0 for m in range(num_lines)] for n in range(5000)]

    # Do predictions for increasingly strict probabilities.
    # Precision equals TP / (TP + FP)
    # Recall = TP Rate TP / (TP + FN)
    print("#Starting on " + str(num_lines) + " leave-one-out trainings.", end="")
    sys.stdout.flush()  # Flush the buffer, so we can see progress.
    for line in range(0, num_lines):
        # Leave-one-out cross-validation, test data is a single row.
        X_test = array(X[line])
        X_test = X_test.reshape((1, X_test.shape[0]))
```

The CSV files which are run through the algorithm give result columns as “Threshold”, ”TP”, “FP”, “FN”, “TN”, “Precision_P”, “Recall_P”.

The analyze function takes the files, number of the algorithm to execute, and column name as input and produce the output files containing the result of the algorithm

```
        # Training data is everything except that row of test data.
        X_train = numpy.delete(X, (line), axis=0)
        y_train = numpy.delete(y, (line), axis=0)
        # Gaussian naive Bayesian.

        if alg_no == 0:
            clf = GaussianNB()
        elif alg_no == 1:
            clf = DecisionTreeClassifier(criterion="entropy", random_state=17)
        elif alg_no == 2:
            clf = LogisticRegression(solver="liblinear", max_iter=1000, tol=0.000000001)
        elif alg_no == 3:
            clf = MLPClassifier(hidden_layer_sizes=(150, 100, 50), max_iter=5000, activation='relu', solver='adam',
                                tol=0.000000001)
        elif alg_no == 4:
            clf = GradientBoostingClassifier()
        elif alg_no == 5:
            clf = HistGradientBoostingClassifier()
        elif alg_no == 6:
            clf = LinearDiscriminantAnalysis()
        elif alg_no == 7:
            clf = RandomForestClassifier(max_depth=10, random_state=17)
        elif alg_no == 8:
            clf = GaussianNB()
            #clf = AdaBoostClassifier(n_estimators=100, random_state=17)
        elif alg_no == 9:
            clf = DecisionTreeClassifier(random_state=17)
        else:
            clf = GaussianNB()
```

Depending on the algorithm number provided the function will choose the algorithm to execute using the if-else statements.

```
        # Use the training data to create a model.
        clf.fit(X_train, y_train)

        # Find the model's predicted probability of each output.
        y_pred = clf.predict(X_test)
        y_prob = clf.predict_proba(X_test)
        # Eye candy.
        if (0 == (line % 10)):
            print(".", end="")
            sys.stdout.flush()  # Flush the buffer, so we can see progress.
        # Is it sufficiently confident to venture a prediction?
        for threshint in range(5000, 10000):
            thresh = threshint / 10000.0  # Threshold goes from 0.5 to 0.999
            my_TP = 0
            my_FP = 0
            my_FN = 0
            my_TN = 0
            if (y_prob[0][0] >= thresh):  # First probability is P = churn, not stay.
                if (y_pred == y_test):
                    my_TP = 1
                    my_FP = 0
                else:
                    my_FP = 1
                    my_TP = 0
            elif (y_prob[0][1] >= thresh):  # Second probability is N = stay, not churn.
                if (y_pred == y_test):
                    my_TN = 1
                    my_FN = 0
                else:
                    my_FN = 1
                    my_TN = 0
```

The for loop is used to iterate through the threshold from 5000 to 10000, providing the threshold from 0.5 to 0.999.

```
            # Should be only one or zero predictions.
            if ((my_TP + my_FP + my_FN + my_TN) > 1):
                print("Error, should be only one prediction, or none, not ",
                      (my_TP + my_FP + my_FN + my_TN))

            # Store that set of results, as an immutable list.
            # print(thresh, line, y_prob[0][0], y_prob[0][1], my_TP, my_FP, my_FN, my_TN)
            single_result = (thresh, line, my_TP, my_FP, my_FN, my_TN)
            all_results[threshint - 5000][line] = single_result

    print(". done.")

    # So how did it go on all those leave-one-out data sets?
    print("#Thresh TP FP FN TN Precision_P    Recall_P")
    for threshint in range(5000, 10000):
        thresh = threshint / 10000.0  # Threshold goes from 0.5 to 0.999
        prev_TP = my_TP
        prev_FP = my_FP
        prev_FN = my_FN
        prev_TN = my_TN
        my_TP = 0
        my_FP = 0
        my_FN = 0
        my_TN = 0
```

```
        for line in range(0, num_lines):
            single_result = all_results[threshint - 5000][line]
            # print(single_result)
            if (1 == single_result[2]):
                my_TP += 1
            elif (1 == single_result[3]):
                my_FP += 1
            elif (1 == single_result[4]):
                my_FN += 1
            elif (1 == single_result[5]):
                my_TN += 1

        # Any different from the previous threshold?
        if ((threshint > 5000) and
                (prev_TP == my_TP) and
                (prev_FP == my_FP) and
                (prev_FN == my_FN) and
                (prev_TN == my_TN)):
            pass  # Don't print anything, same as previous line.
        else:
            # If there were any predictions, calculate precision and TP rate.
            if ((my_TP + my_FP) > 0):
                my_precis = my_TP / (my_TP + my_FP)
            else:
                my_precis = 0

            if ((my_TP + my_FN) > 0):
                my_recall = my_TP / (num_positives)  # Out of all positive cases.
            else:
                my_recall = 0
```

```

            print(str(thresh), "  ", my_TP, my_FP, my_FN, my_TN,
                  "  ", my_precis, my_recall)
            arr = [str(thresh),my_TP,my_FP,my_FN,my_TN,my_precis,my_recall]
            data_to_append = {}
            for i in range(len(final_df.columns)):
                data_to_append[final_df.columns[i]] = arr[i]
            final_df = final_df.append(data_to_append, ignore_index=True)

    # All finished, repeat the row of headers.
    print("#Thresh TP FP FN TN Precision_P    Recall_P")
    createFile(final_df,"OUTPUT" + str(column_Name) + str(data))

```

Another for loop is used to store all the output of the algorithm to an array which is then appended into a data frame which is finally used to create an output file.

Differences between the alpha, beta, and final releases:

ALPHA-Release:

The user stories for the alpha release where:

- **Load all the files**
- **Merge the file based on the common attribute**
- **Clean the file and fill in missing values**

For the alpha-release, a python code was created using PyCharm which loads all the CSV files. The raw data has got many missing and null values. So, we had to clean the data by replacing missing values with ‘0’s.

The next task was to select columns that are required to predict information. The python code loads the files and then merges the columns. The merged data frame is then converted to a CSV file and stored in the system.

```
305  def mergeFiles(loadedFiles):
306      asDF = pd.merge(loadedFiles[0], loadedFiles[2], on=[MERGECOLUMN_1, MERGECOLUMN_2],
307                      how='outer')  # first merging the admission file and the symptoms file
308      aspDF = pd.merge(asDF, loadedFiles[3], on=[MERGECOLUMN_2], how='outer')  # Then merging the new file with patients
309      aspdDF = pd.merge(aspDF, loadedFiles[1], on=[MERGECOLUMN_3])  # Now merging the new file with diagnoses
310      return aspdDF
```

BETA-Release:

The user stories for the alpha release where:

- **Updating the code's efficiency**
- **Creating different files each based on a different primary disease (Only one row for a patient)**
- **Creating different files each based on gender(Only one row for a patient)**
- **Creating different files each based on insurance (Only one row for a patient)**

The merged file contained some additional columns which are not mandatory for this project and resulted in a long-running time. So, we have given specific names of the

columns which are only required for the project. And thereby, dropping all the other non-mandatory columns. This is how we increased the coding efficiency and decreased running time.

The next user story for beta-release was to split files based on primary disease, where a patient will only have one row as the patient entry. By splitting, a file is generated for every disease containing all the patients who have that disease. The third and fourth user story is similar to the splitting based on the primary disease. So, we can use a general function for splitting files. The function which splits files is 'def split(column_Name, Data_File)' shown below:

```
def split(column_Name, Data_file):
    # listing unique data
    data_list = Data_file[column_Name].unique()

    # creating different files each for each unique
    algo_num = [x for x in range(10)]
    for data in data_list:
        new_Data = Data_file.loc[Data_file[column_Name] == data]
        del new_Data[column_Name]
        print('*' + str(data) + '***')
        print(new_Data)
        no_of_ones = list(new_Data.EXPIRE_FLAG).count(1)
        no_of_zeros = list(new_Data.EXPIRE_FLAG).count(0)
        print("No of Ones " + str(no_of_ones))
        print("No of Zeros " + str(no_of_zeros))
        if len(new_Data) > 200 and len(new_Data) < 700 and no_of_ones > 0 and no_of_zeros > 0:
            print("**  " + data + " ***")

            for n in algo_num:
                print("\n***  " + ALGO_NAMES[n] + " **\n")
                analyse(new_Data, n, column_Name,data)
        createFile(new_Data, str(column_Name) + str(data))
```

FINAL-Release:

The final release should contain all the user requirements and the user stories for the final iteration is as below:

- **Preparing the files to be fed as an input to the data mining algorithm.**
- **A new python program containing multiple data mining algorithms which load all the CSV files created by the first program and perform algorithm**
- **Leave one out cross-validation is to be done on the CSV files**
- **The threshold of the cross-validation can be changed between 0.5 and 1.**

The merged files and the split files had non-numeric values and symbols in their cells. To run through the algorithm, we developed a code that changes all non-numeric values to numbers. This helps the algorithm to work perfectly and we get the desired output.

We developed code to perform various algorithms and then we determine which algorithm works best for this data. The algorithms we used are Gaussian NB, Decision Tree Classifier, Gradient Boosting Classifier, Logistic Regression, MLP Classifier, Random Forest Classifier, Hist Gradient Boosting Classifier, Linear Discriminant Analysis.

CLIENT SIGNED ACCEPTANCE OF THE FINAL-RELEASE

Client Acceptance of Final Release

Completed Work

Sl.no	Title	Iteration
1	Python Program v0.7	A new python program containing multiple data mining algorithms which loads all the CSV files created by the first program and perform algorithm
2	Python Program v0.8	Leave one out cross-validation is to be done on the csv files
3	Python Program v0.9	Threshold of the cross validation can be changed between 0.5 and 1

Client Feedback:

The use of several algorithms from the Scikit-Learn library is done nicely, it goes through them on each of the CSV files, to measure an algorithm's accuracy on a particular file, for various values of the threshold, which ranges from 50% to 100%. There might be some CSV files where every row has the same output (e.g., everyone with the disease did not get readmitted, or suchlike), so that certain algorithms don't give a sensible answer, but that's perfectly okay when there are so many combinations and permutations to produce all those CSV files.

The cross-validation is artfully done. Of course the software will take time to run, because for a CSV file with (say) 500 rows, each algorithm will produce 500 slightly different models, one for each row left out to become the single point of unseen test data. So larger files will take more time, not just because there is more data, but because leave-one-out cross-validation takes longer for bigger files.

The threshold value iterates from 50% to 100%, and only prints out a new line when the result changes, which reduces the size of the resulting output.

This software will allow research work on which diseases are predictable by which algorithms, and conversely, which diseases cannot be accurately predicted with current technology. So congratulations on all your good work, and rest satisfied after a job well done.

CHANGES FROM ITERATION-3 ARE APPROVED BY THE CLIENT.

The results of the algorithm could be saved as text or chat, either way, the client was satisfied. At first, our plan included chart results but, due to the limited time, we had to drop the chart-based result for a later period.

The merged files and the split files had non-numeric values and symbols in their cells. To run through the algorithm, we developed a code that changes all non-numeric values to numbers. This helps the algorithm to work perfectly and we get the desired output.

CHARTS AND ANALYSIS

User Stories

The user stories and all the analysis is maintained using Microsoft Excel-based on different iteration. The percentage completed, average velocity, total estimated time frame, all are shown in detail. The user stories which are postponed outside the iteration are given in the parking lot. The user story which is dropped is given in the dropped list.

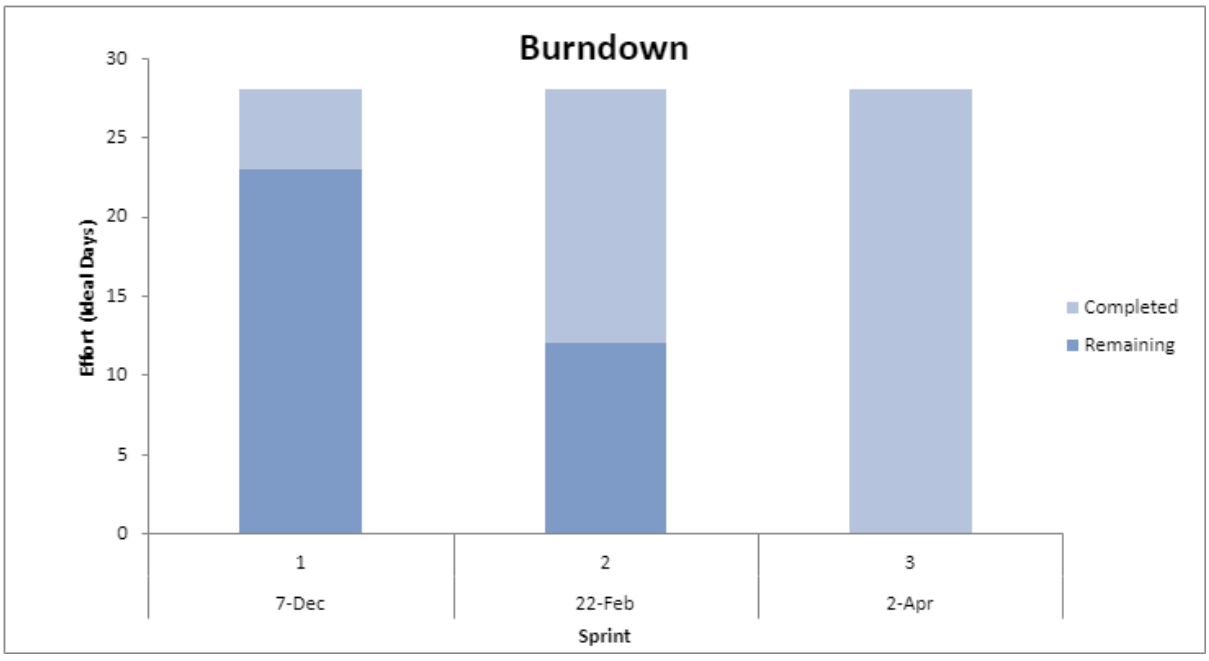
Sprint	Story	Estimate
1	Dec-7	
	Load all the files	2
	Merge the file based on the common attribute	2
	Clean the file and fill missing values	1
2	Feb-22	
	Updating the code's efficiency	1
	Creating different files each based on a different primary disease (Only one row for a patient)	5
	Creating different files each based on gender(Only one row for a patient)	3
	Creating different files each based on insurance (Only one row for a patient)	2
3	Apr-2	
	A new python program containing multiple data mining algorithms which load all the CSV files created by the first program and perform algorithm	3
	Leave one out cross-validation is to be done on the CSV files	2
	The threshold of the cross-validation can be changed between 0.5 and 1	2
	Preparing the files to be fed as an input to the data mining algorithms	5
	Release Feb-19	
	Total	28
	Completed	28
	Remaining	0
	% Complete	100%
	Avg Velocity	5
	Estimated # Iterations Remaining	0
	Parking Lot	
	Add details of the first blood test in the above 3 files	1

Dropped		
	Stuff we didn't need to Splitting the based on age	6
	Code to create a chart based on the result of the cross-validation	2

Burn-down charts

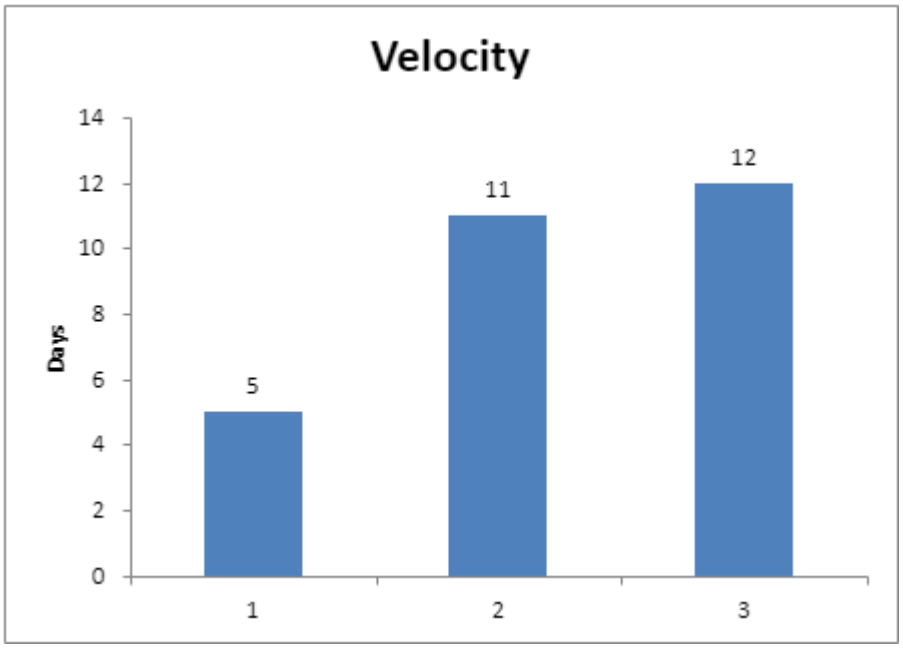
The burn-down chart is a bar graph that is based on the iteration. It shows a graphical representation of the first and second iteration.

Start Date	7-Dec	22-Feb	2-Apr
Sprint No	1	2	3
Total Effort	28	28	28
Velocity	5	11	12
Remaining	23	12	0
Completed	5	16	28
% complete	18%	57%	100%



Velocity charts

Velocity charts show the pace at which the project progresses. Both the iterations are displayed here using the bar graph. Any changes made in excel will reflect in the graphs.



PROJECT MAINTENANCE ICT INFRASTRUCTURE

DEVELOPMENT ENVIRONMENT

PyCharm

PyCharm is one of the best, if not the best, full-featured, dedicated, and versatile IDEs for Python development. It offers a ton of benefits, saving you a lot of time by helping you with routine tasks.

PROGRAMMING LANGUAGES

Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting, and was discontinued version 2.7.18 in 2020. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3.

Python consistently ranks as one of the most popular programming languages.

Pandas Library

Built on top of the Python programming language, pandas is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool. Working using pandas makes data analysis much faster and more effective. Pandas support all the major file formats such as CSV, xlsx file, SQL database.

In this project, we used the panda's library to split files, merge multiple files based on primary disease,

Scikit Library

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

In this tutorial, we will learn to code python and apply Machine Learning with the help of the scikit-learn library, which was created to make doing machine learning in Python easier and more robust.

To do this, we'll be using the Sales_Win_Loss data set from IBM's Watson repository. We will import the data set using pandas, explore the data using pandas methods like head(), tail(), dtypes(), and then try our hand at using plotting techniques from Seaborn to visualize our data.

Then we'll dive into scikit-learn and use preprocessing.LabelEncoder() in scikit-learn to process the data, and train_test_split() to split the data set into test and train samples. We will also use a cheat sheet to help us decide which algorithms to use for the data set. Finally, we will use three different algorithms (Naive-Bayes, LinearSVC, K-Neighbors Classifier) to make predictions and compare their performance using methods like accuracy_score() provided by the scikit-learn library. We will also visualize the performance score of different models using scikit-learn and Yellowbrick visualization.

NumPy Library

NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

NumPy stands for Numerical Python. In Python, we have lists that serve the purpose of arrays, but they are slow to process.NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

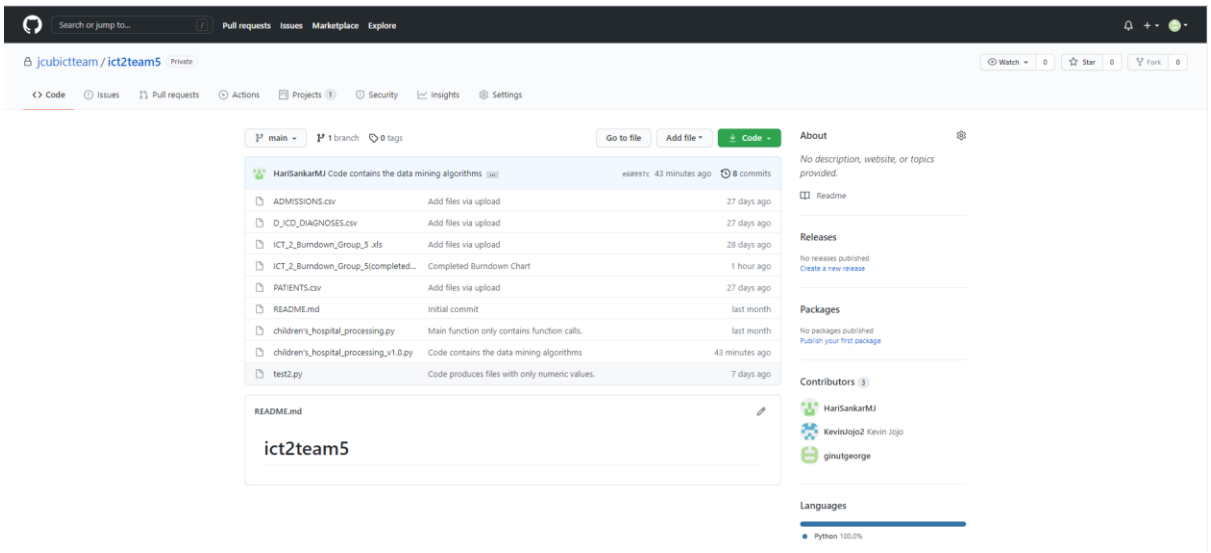
The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behavior is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also, it is optimized to work with the latest CPU architectures.

NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++. The source code for NumPy is located at this GitHub repository https://github.com/numpy/numpy

SOURCE CODE REPOSITORIES (CONFIGURATION MANAGEMENT)

GitHub - https://github.com/jcubictteam/ict2team5



GitHub is a platform for software developers to store and share their projects, maintain versions of the code, source code management functionality, and several other features. GitHub is free software for using its basic features but, for more advanced features and to use for enterprise services, subscription is required.

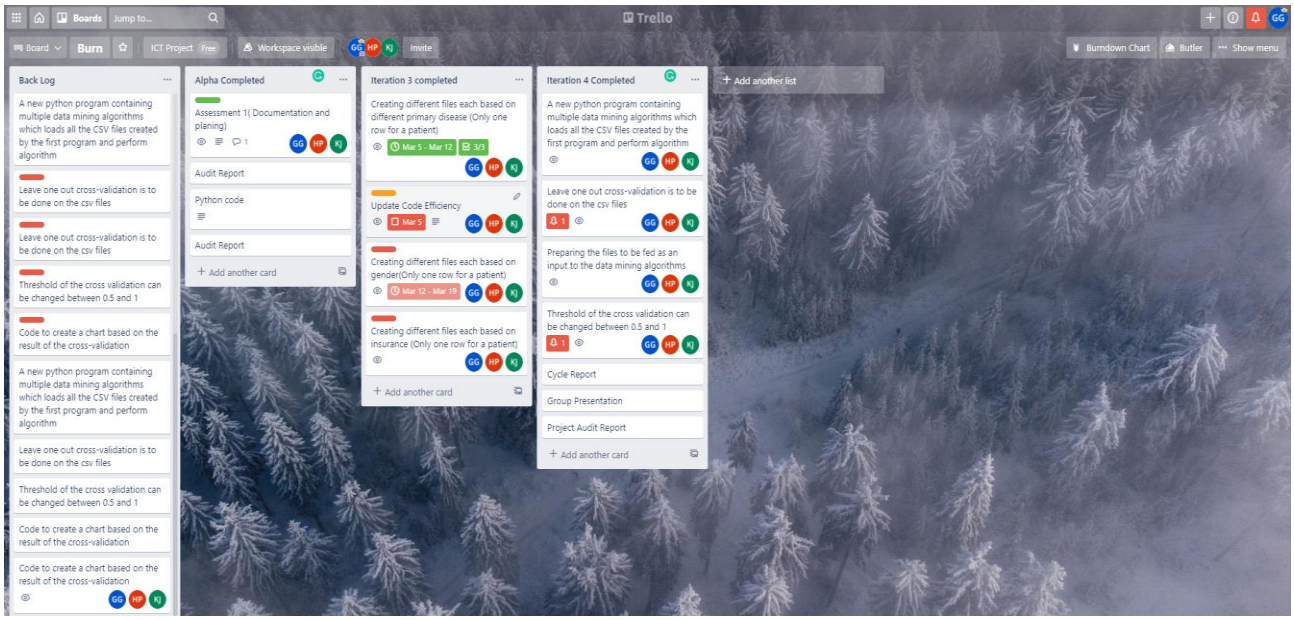
Each teammate has access to that project in which they are working and modify changes. Each member can upload files and the time and date are also stored thereby, any errors can be rechecked since versions are stored.

PROJECT COLLABORATION TOOLS

TRELLO

Trello Link –

https://trello.com/invite/b/DUEMd8YJ/160bf672c2df283da6f6e198badd8583/burn



Trello is a tool based on the cloud and uses the project management Kanban technique. According to the Kanban system, all project operations are seen in a single landscape visible to all project team members. Trello offers users the option of visually organizing projects into committees, the division into groups, and dividing into activities into groups. The user-friendly interface from Trello makes it suitable for a wide range of users from people who manage personal tasks like home repairs to companies who run many big projects and teams. Trello provides varying service levels at different cost points to satisfy the demands of the clients. We have used the free version of Trello.

Registered users can establish unlimited board numbers and designate 1 board per project. Users can subsequently assign to each board several task groups (lists) and assign sub-cards to each list. Users can create cards by manually adding or copying and

pasting existing Microsoft Word or Excel text lists. Users can create a card for each text line with the latter procedure or create a single card with multiple text lines. The to-do lists that appear only when cards are expanded can be further granulated. Users are also allowed to upload annexes and add card comments.

Trello is an excellent tool for project management that can help users to carry out projects in due course. Trello allows users to communicate with board members by using the commenting feature and by assigning members to cards. It is particularly useful to be able to allocate due dates and include them in a personal calendar. In addition, Trello enables users to get a visual snapshot of project progress via the use of labels and due date notifications.

Team Communication

WhatsApp

WhatsApp Link- https://chat.whatsapp.com/J9NrHKMKbpzJn4t2SmJs8b

We have used WhatsApp as the main communication app between all teammates and every week on Friday at 2 pm all team members had team meetings and discussed the progress and discussed the timeline.


5
ICT TEAM 5
Hari, Kevin, You
Team Meeting for Project Audit
Thursday Morning
10:56 am
Onedrive Link For Project Audit Document : https://myjcuedu-my.sharepoint.com/:w:/g/personal/ginu_tholoorparampilgeorge_my_jcu_edu_au/EYAc8XoEbvpBh2ZMvva6t14BRGVu5wHPqy_oyK2-n4MUNg?e=KnOVtO
10:58 am
Today
Hi Kevin,
Please send me burn down chart
Thanks
Ginu
10:33 am
Kevin Jcu
I've already mailed you
10:33 am
10:34 am
Hari JCU IT
I uploaded the current version of the code
11:01 am

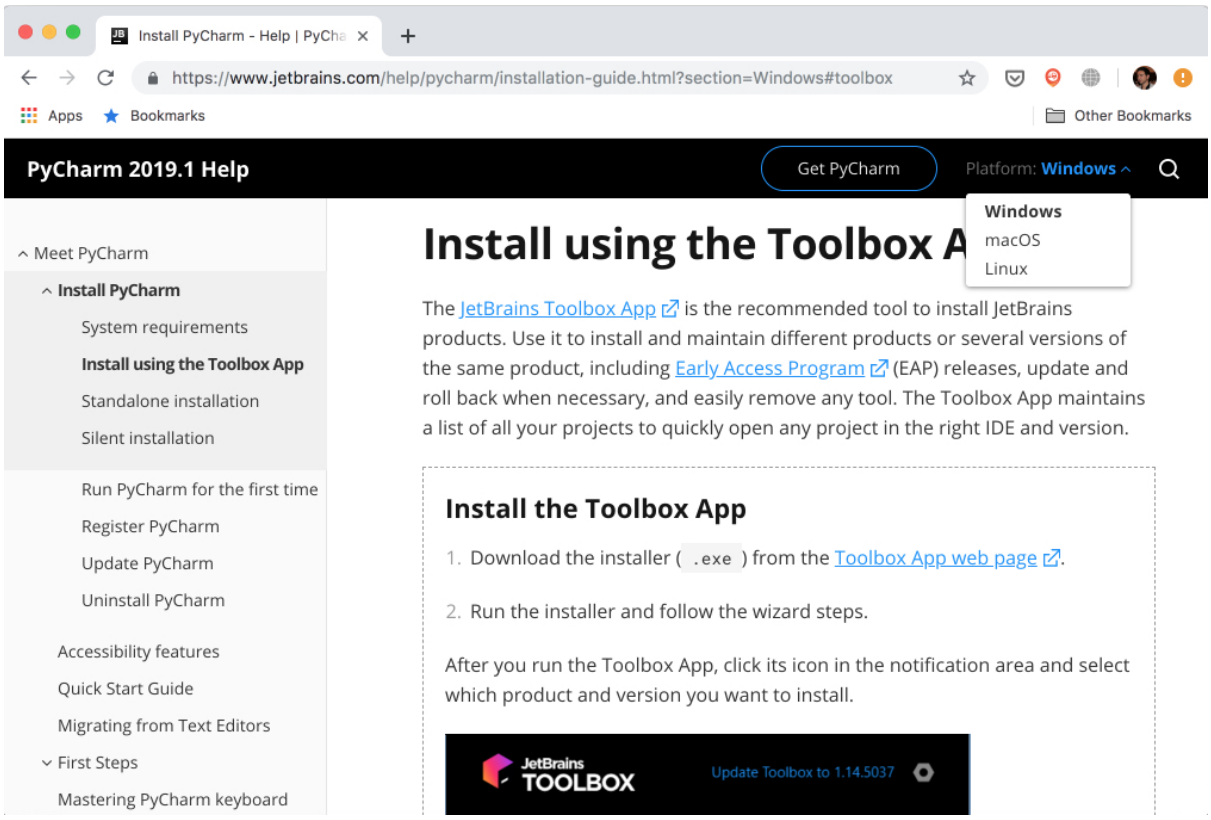
CLIENT TRAINING DOCUMENT, AND PROCEDURES

Installing PyCharm

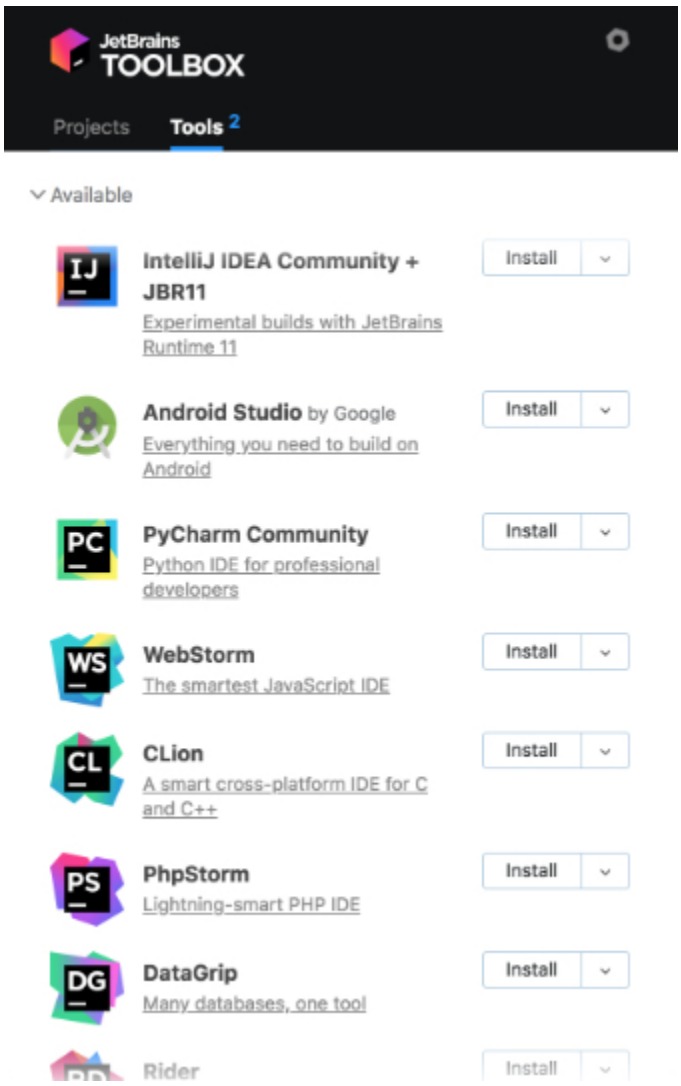
PyCharm Community Edition 2019.1 as it's free and available on every major platform. Only the section about the professional features will use PyCharm Professional Edition 2019.1.

The recommended way of installing PyCharm is with the JetBrains Toolbox App. With its help, you'll be able to install different JetBrains products or several versions of the same product, update, roll back, and easily remove any tool when necessary. You'll also be able to quickly open any project in the right IDE and version.

To install the Toolbox App, refer to the documentation by JetBrains. It will automatically give you the right instructions depending on your OS. In case it didn't recognize your OS correctly, you can always find it from the drop-down list on the top right section:

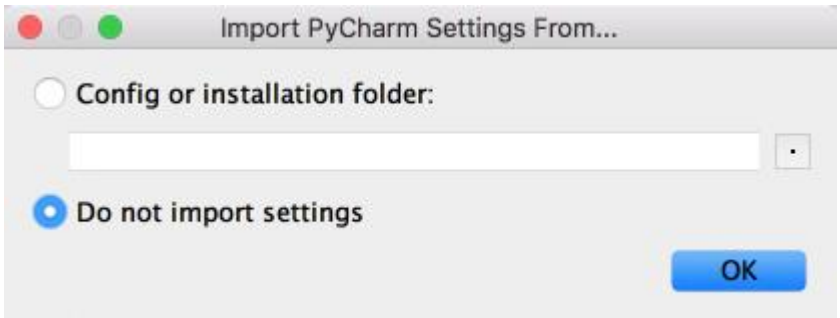


After installing, launch the app and accept the user agreement. Under the *Tools* tab, you'll see a list of available products. Find PyCharm Community there and click *Install*:

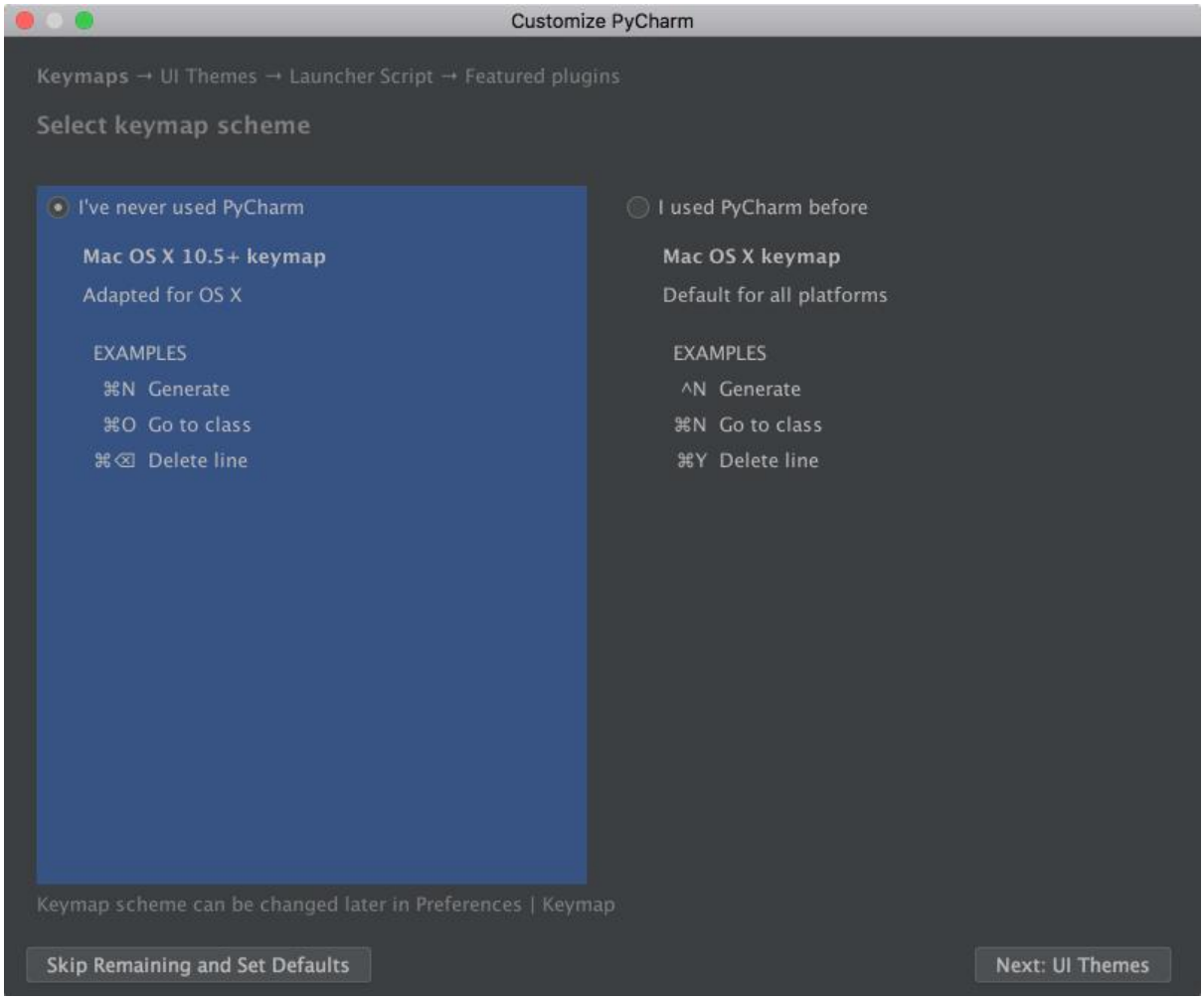


Voilà! You have PyCharm available on your machine. If you don't want to use the Toolbox app, then you can also do a stand-alone installation of PyCharm.

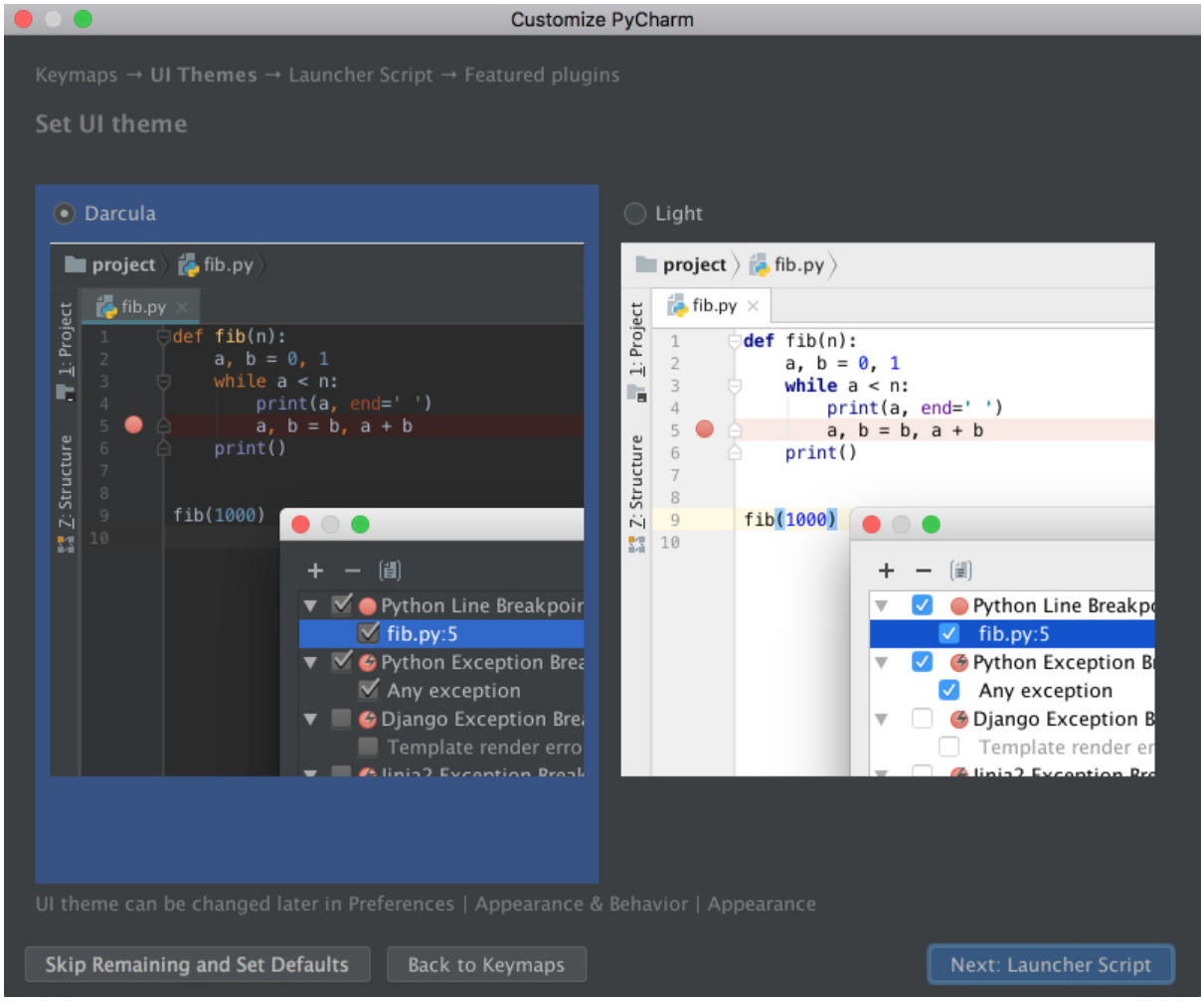
Launch PyCharm, and you'll see the import settings popup:



PyCharm will automatically detect that this is a fresh install and choose *Do not import settings* for you. Click *OK*, and PyCharm will ask you to select a keymap scheme. Leave the default and click *Next: UI Themes* on the bottom right:



PyCharm will then ask you to choose a dark theme called Darcula or a light theme. Choose whichever you prefer and click *Next: Launcher Script*:



I'll be using the dark theme Darcula throughout this tutorial. You can find and install other themes as plugins, or you can also import them.

On the next page, leave the defaults and click *Next: Featured plugins*. There, PyCharm will show you a list of plugins you may want to install because most users like to use them. Click *Start using PyCharm*, and now you are ready to write some code!

Editing an Existing Project in PyCharm

These single file projects are great for examples, but you'll often work on much larger projects over a longer period. In this section, you'll take a look at how PyCharm works with a larger project.

To explore the project-focused features of PyCharm, you'll use the Alcazar web framework that was built for learning purposes. To continue following along, clone the repo locally:

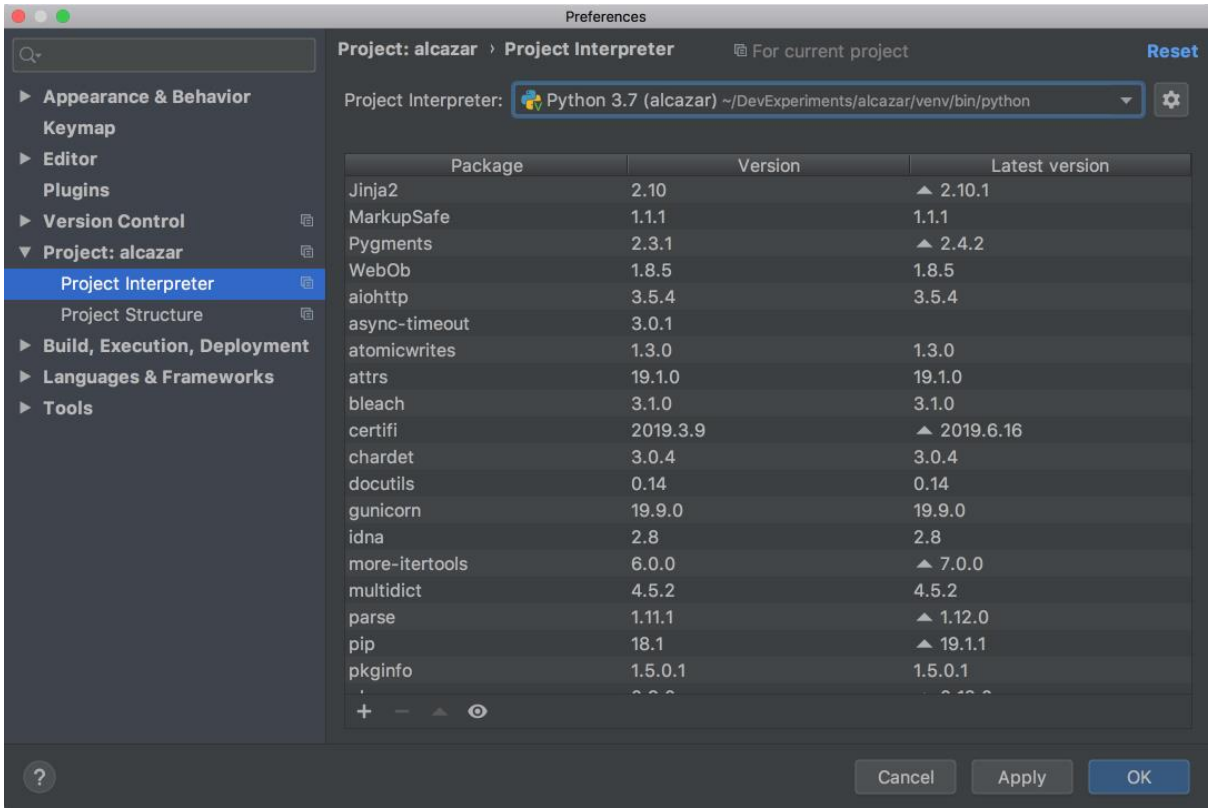
Once you have a project locally, open it in PyCharm using one of the following methods:

- Click *File-Open* on the main menu.
- Click *Open* on the Welcome Screen if you are there.

After either of these steps, find the folder containing the project on your computer and open it.

If this project contains a virtual environment, then PyCharm will automatically use this virtual environment and make it the project interpreter.

If you need to configure a different virtualenv, then open *Preferences* on Mac by pressing Cmd + , or *Settings* on Windows or Linux by pressing Ctrl + Alt + S and find the *Project: ProjectName* section. Open the drop-down and choose *Project Interpreter*:



Choose the virtualenv from the drop-down list. If it's not there, then click on the settings button to the right of the drop-down list and then choose *Add…*. The rest of the steps should be the same as when we were creating a new project.

Future Improvements and maintenance

This project does not require frequent maintenance. But the dropped user stories will be done soon. The code for chart-based results will be updated as an update to the current code.

Also, the coding efficiency will be increased and reduce the running time.

TEST CASES

Testing conditions	Pre-conditions	Test-cases	Desired result	Actual result
Splitting based on diseases.	Data frame containing the merged files, the name of the diseases column	Calling the function split and passing the values 'DISEASES' and the merged dataset.	Individual files for each of the diseases with all the details of the patients are created	Individual files for each disease with all the patient details are created and printed
Splitting based on gender.	Data frame containing the merged files, the name of the gender column	Calling the function split and passing the values 'GENDER' and the merged dataset.	Individual files for each gender with all the details of the patients are created.	Individual files for each gender with all the patient details are created and printed.
Splitting based on different insurance	Data frame containing the merged files, the name of the insurance column	Calling the function split and passing the values 'INSURANCE' and the merged dataset.	Individual files for each of the insurance types with all the details of the patients are created.	Individual files for each type of insurance with all the patient details are created and printed.
Increasing the efficiency of the code.	The initial code had a run time of around 3 minutes.	Trimming the dataset of unnecessary columns	Getting a runtime under 60 seconds.	Getting a runtime under 25 seconds.
A new function containing multiple data mining algorithms loads all the CSV files created by the first program	Sample CSV file ready to be input to the data mining algorithm and function containing the datamining algorithm.	Giving the sample file as input the data mining function.	Output file containing the output of the data mining algorithm in a CSV file.	Output file containing the output of the data mining algorithm in a CSV file.

and performs the algorithm.				
Preparing the files to be fed as an input to the data mining algorithms. Leave one out cross-validation is to be done on the CSV files.	All files that needed to be the input of the data mining function and the function for cleaning the data and replacing the string values with numbers.	The processed file is used as an input to the data mining algorithm and check if any type errors are generated.	The input files are accepted without generating any type of error.	The input files are accepted without generating any type of error.
The threshold of the cross-validation can be changed between 0.5 and 1.	Function to perform the data mining algorithm and the input files.	Checking the output file for the different threshold results.	Results for different thresholds should be present in the output file.	The result for different thresholds is present in the output file.