Assignment #2- Git, GitHub, ML

Step 1:

Install git in your local machine (If it is already done, skip this step)

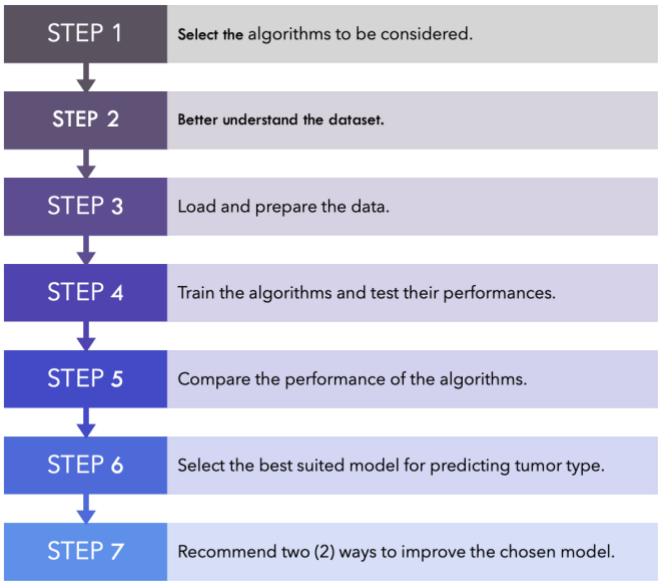
SKIPPED

Note: This is the third time I will be working with Wisconsin Breast Cancer Dataset. Therefore, the following steps were carried out using the code I wrote when I worked

---

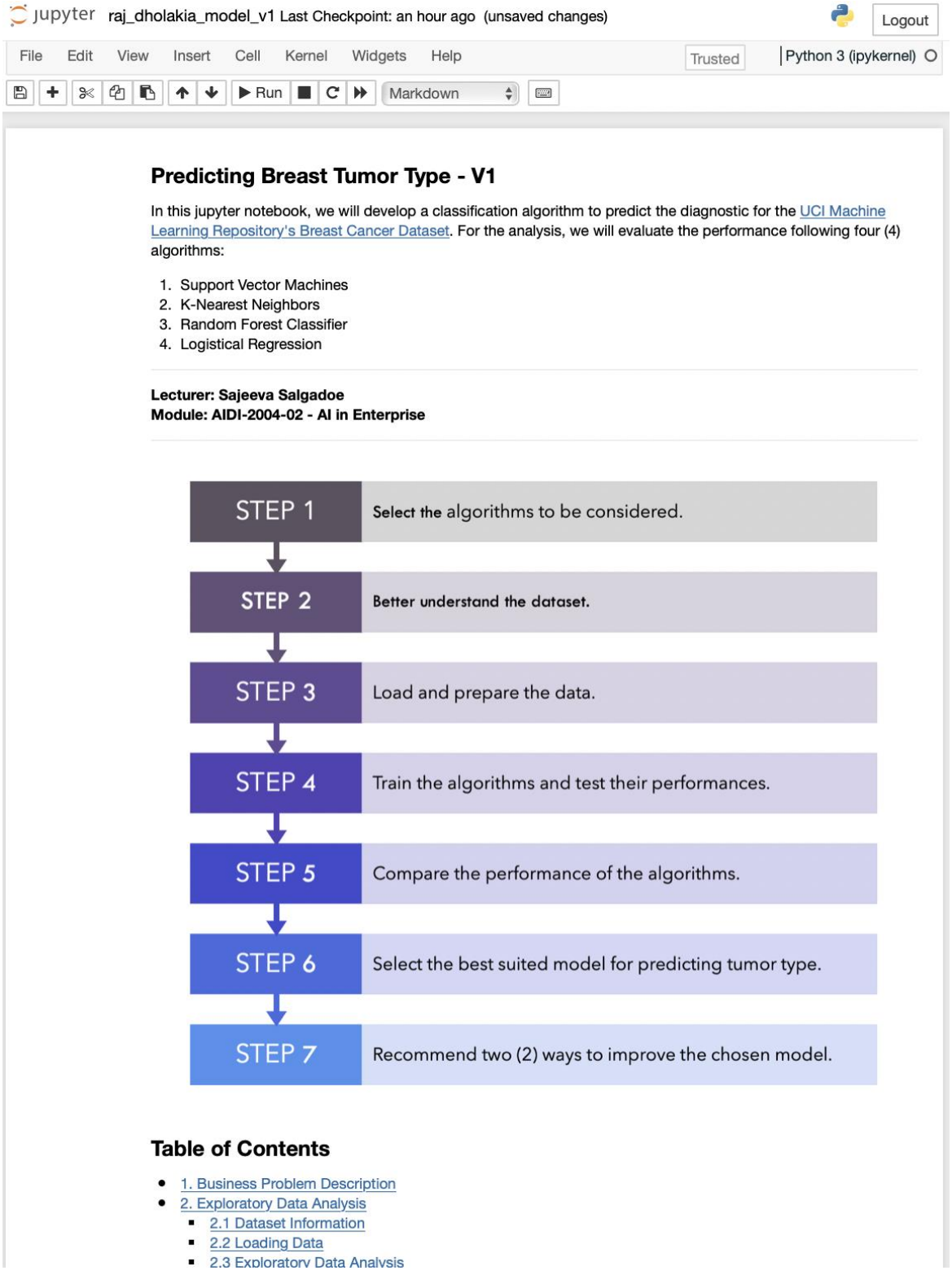
Step 2:

Build a ML model for Breast Cancer Wisconsin (Diagnostic) Data set in Jupyter notebook.
https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)



Step 6 and Step 7, in the above picture, will be carried at a later stage in this document.

Step 3:
Please provide screenshots for various stages of the design process (importing data, training, evaluation …)



2.2 Loading Data

In [1]:
```
# Load required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix, auc

import os
import pathlib
from helper_functions import metric_evaluation, learning_curve, full_model_evaluatic
# from helper_functions import plot_conf_mat, evaluate_model
```

In [2]:
```
# Define location of the data
data_dir = '../data'
filename = 'dataset.csv'
data_path = os.path.join(data_dir, filename)

if pathlib.Path(data_path).exists():
    print(f"File {filename} found.")
else:
    raise FileNotFoundError('No file found at the location defined.')
```

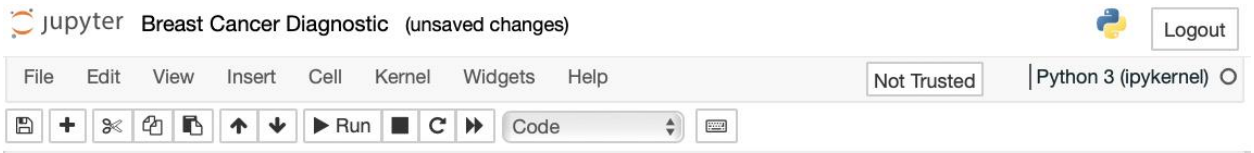
```
File dataset.csv found.
```

In [3]:
```
# Load data into a pandas DataFrame
data = pd.read_csv(data_path)
data.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	87139402	B	12.32	12.39	78.85	464.1	0.10280
1	8910251	B	10.60	18.95	69.28	346.4	0.09688
2	905520	B	11.04	16.83	70.92	373.2	0.10770
3	868871	B	11.28	13.39	73.00	384.8	0.11640
4	9012568	B	15.19	13.21	97.65	711.8	0.07963

5 rows × 32 columns



3. Data Preparation

In [15]:

```python
# Define x and y variables for CMS prediction
x = data.drop('diagnosis', axis=1).to_numpy()
y = data["diagnosis"].replace({"M": 1, "B": 0})

# Splitting data into train and test datasets
x_train, x_test, y_train, y_test = train_test_split(x, y, stratify=y, test_size=0.2,

# Scaling the data
sc = StandardScaler()
x_train_scaled = sc.fit_transform(x_train)
x_test_scaled = sc.transform(x_test)

# Created scaled x, to use cross-validated
x_scaled = sc.fit_transform(x)
```

4. Development of Models

In this section, we will train and test the performance of the model based on the **cross validated scores** of the following metrics:

1. `accuracy`
2. `precision_weighted`
3. `recall_weighted`
4. `f1_weighted`
5. `roc_auc`

4.1 Cross Validation

- A 5-fold cross validation score, first splits the data into five (`5`) parts.
- It then, considers one (1) part to be the test set and trains the model on the rest of the data. It repeats this for all the five (5) parts.

> **Note**: in this analysis, an average of the cross validated scores is taken into consideration. One could also look at individual values to observe if the performance of a particular algorithm is consistent on all the five (5) parts/folds.

- Using this method will give us a better understanding of which algorithms perform better.
- The Brier Score uses probabilities and the expected values to calculate the mean squared error. The algorithms support probabilities.

jupyter Breast Cancer Diagnostic (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

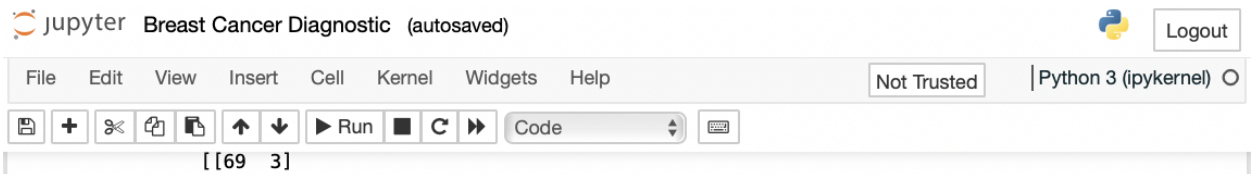
Run Code

4.2 Model Training

In [16]:

```
models = {}

for name,method in [('Logistic Regreesion', LogisticRegression()),
                    ('KNN', KNeighborsClassifier()),
                    ('Random Forest', RandomForestClassifier()),
                    ('Decision Tree', DecisionTreeClassifier())]:
    # Train the model
    method.fit(x_train_scaled, y_train)

    # Make predictions
    predictions = method.predict(x_test_scaled)

    # Evaluate the performance
    target_names=['benign','malignant']
    print(f'\n-------------  {name}  ---------------')
    print(confusion_matrix(y_test, predictions))
    print(classification_report(y_test, predictions, target_names=target_names))


    # Save the models
    models.update({name: method})
```

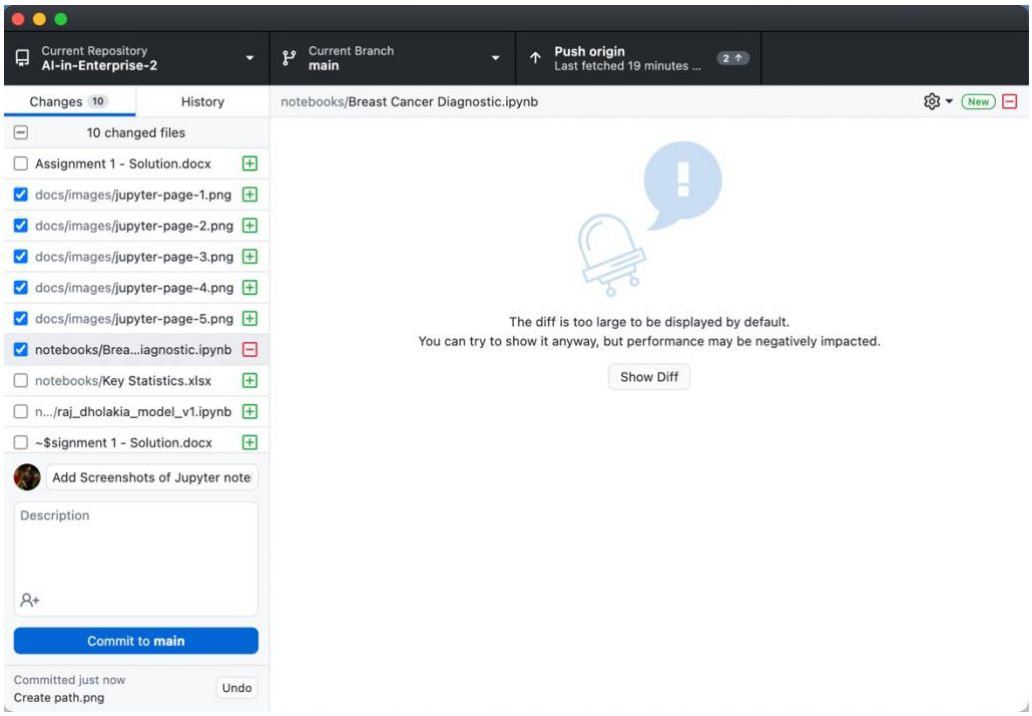
```
-------------  Logistic Regreesion  ---------------
[[69  3]
 [ 0 42]]
              precision    recall  f1-score   support

      benign       1.00      0.96      0.98        72
   malignant       0.93      1.00      0.97        42

    accuracy                           0.97       114
   macro avg       0.97      0.98      0.97       114
weighted avg       0.98      0.97      0.97       114


-------------  KNN  ---------------
[[71  1]
 [ 1 41]]
              precision    recall  f1-score   support

      benign       0.99      0.99      0.99        72
   malignant       0.98      0.98      0.98        42

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

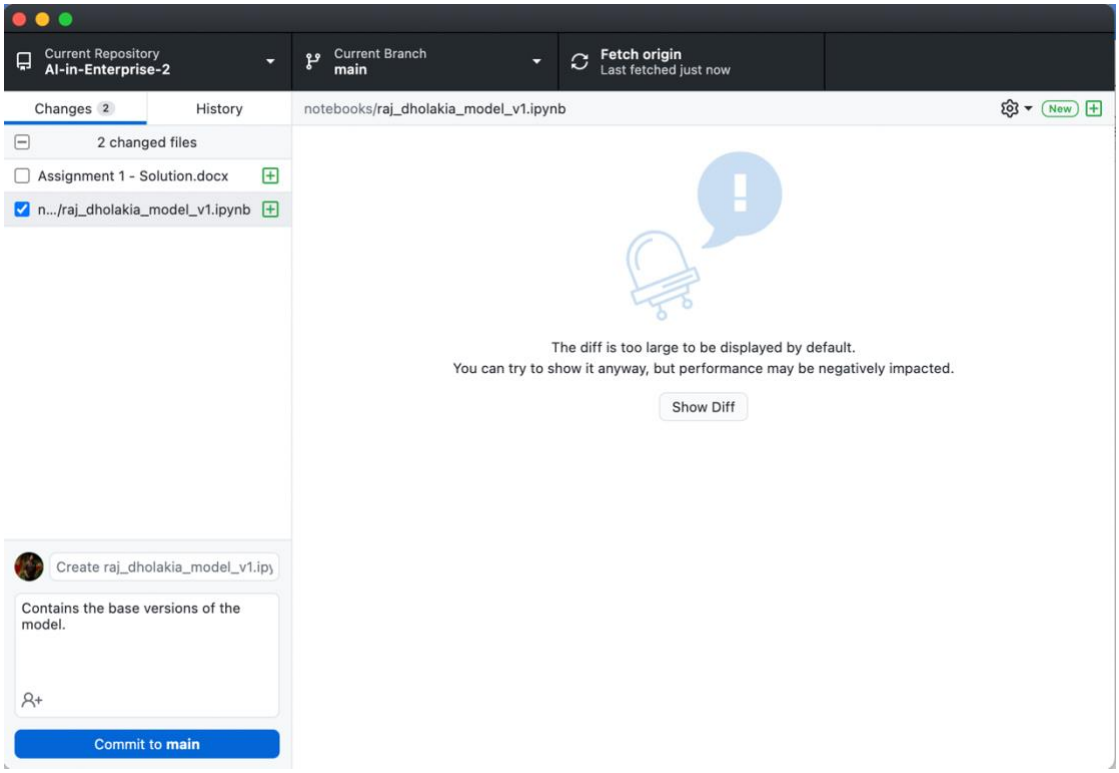



```
[[69  3]
 [ 0 42]]
              precision    recall  f1-score   support

      benign       1.00      0.96      0.98        72
   malignant       0.93      1.00      0.97        42

    accuracy                           0.97       114
   macro avg       0.97      0.98      0.97       114
weighted avg       0.98      0.97      0.97       114


-------------  KNN  ---------------
[[71  1]
 [ 1 41]]
              precision    recall  f1-score   support

      benign       0.99      0.99      0.99        72
   malignant       0.98      0.98      0.98        42

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114


-------------  Random Forest  ---------------
[[70  2]
 [ 1 41]]
              precision    recall  f1-score   support

      benign       0.99      0.97      0.98        72
   malignant       0.95      0.98      0.96        42

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114


-------------  Decision Tree  ---------------
[[69  3]
 [ 1 41]]
              precision    recall  f1-score   support

      benign       0.99      0.96      0.97        72
   malignant       0.93      0.98      0.95        42

    accuracy                           0.96       114
   macro avg       0.96      0.97      0.96       114
weighted avg       0.97      0.96      0.97       114
```

Step 4:

Upload your model (Python script, let's called it <yourname>_model_v1) to GitHub. Provide screenshot of all your git commands and your command prompt showing success of commit of your model files in the remote host.



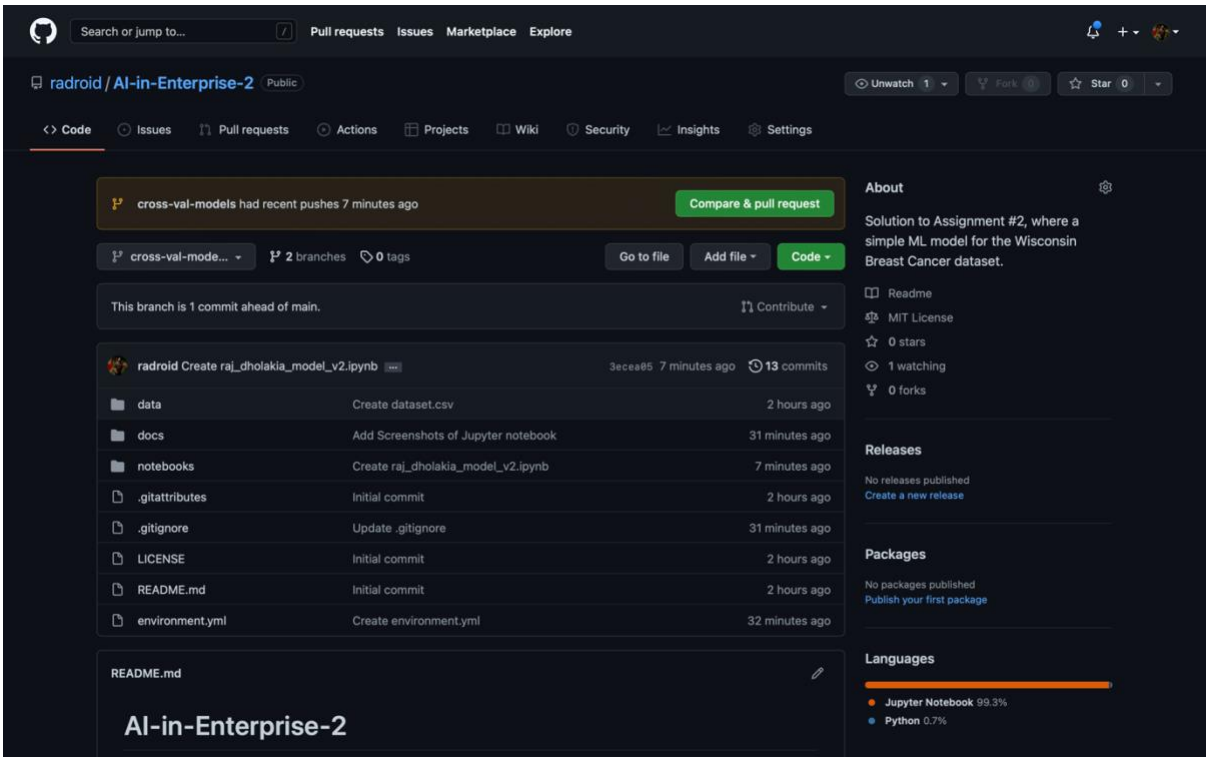


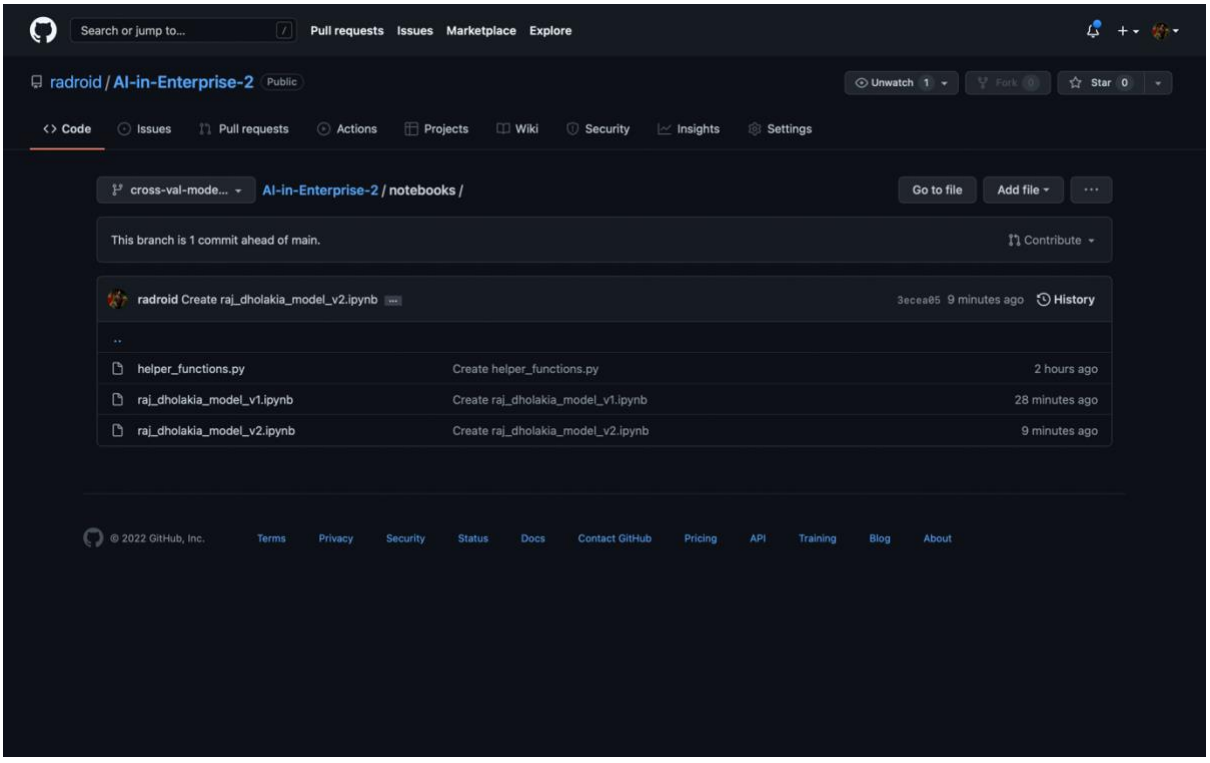
Step 5:

Create a branch in your repo and upload another ML model (may be using a different algorithm and named the file: <yourname>_model_v2) of your choice for the same dataset into that branch.

Step 6:

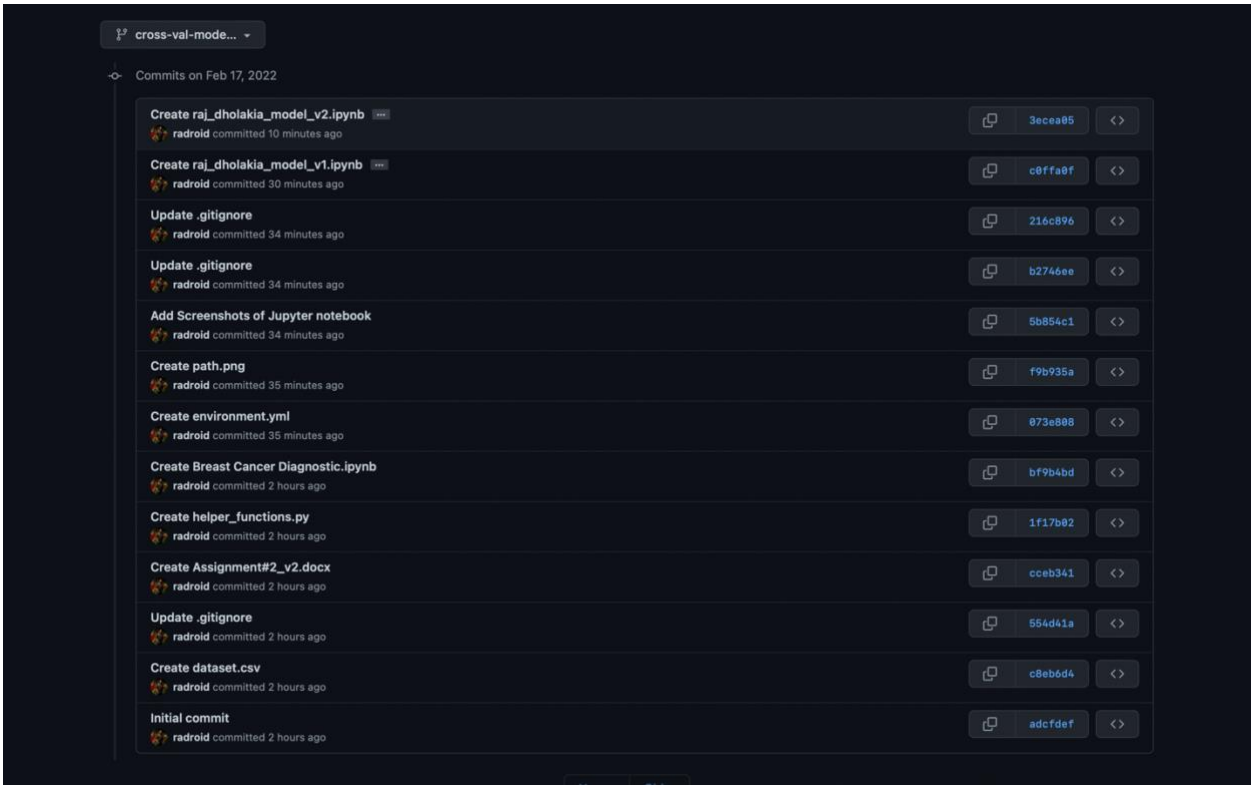
Navigate to your newly created branch and provide screenshot showing status of your repo.





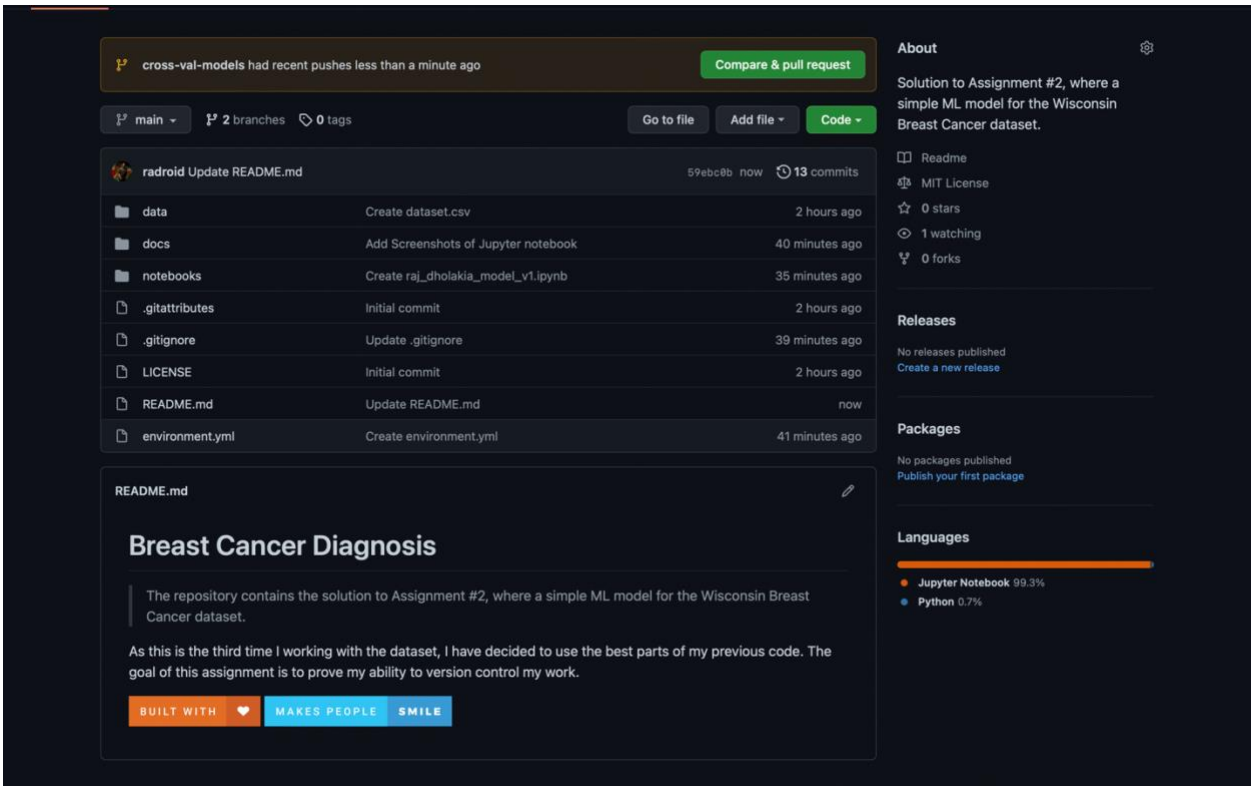
Step 7:

Provide a screenshot showing your log of activities and perform your final commit.



Step 8:

Provide a description of your program in the README.md file.



Step 9:

Make your repo public and share the link of your repo for check.

Link to the repository: [AI-in-Enterprise-2]

THANK YOU FOR READING!