Assignment-1

January 17, 2025

1 Introduction to Machine Learning

1.1 Assignment 1: Machine Learning Fundamentals

You can't learn technical subjects without hands-on practice. The assignments are an important part of the course. To submit this assignment you will need to make sure that you save your work before closing Jupyter notebook and submit your ipynb file on Blackboard

1.1.1 Assignment Learning Goals:

By the end of the module, students are expected to:

- Explain motivation to study machine learning.
- Differentiate between supervised and unsupervised learning.
- Differentiate between classification and regression problems.
- Explain machine learning terminology such as features, targets, training, and error.
- Use DummyClassifier/ Dummy Regressor as a baseline for machine learning problems.
- Explain the `.fit()` and `.predict()` paradigm and use `.score()` method of ML models.

Any place you see `...`, you must fill in the function, variable, or data to complete the code. Substitute the `None` with your completed code and answers then proceed to run the cell!

Note that some of the questions in this assignment will have hidden tests. This means that no feedback will be given as to the correctness of your solution. It will be left up to you to decide if your answer is sufficiently correct. These questions are worth 2 points.

[1]:
```
# Import libraries needed for this lab
from hashlib import sha1

import altair as alt

import graphviz
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import test_assignment1 as t
from IPython.display import HTML
from sklearn import tree
from sklearn.dummy import DummyClassifier
```

```
from sklearn.model_selection import cross_val_score, cross_validate,␣
 ↪train_test_split
from sklearn.tree import DecisionTreeClassifier
```

1.2 1. About Machine Learning

Question 1.1 {points: 1}

Which of the following is **NOT** a type of Machine Learning?

To answer the question, assign the letter associated with the correct answer to a variable in the code cell below:

A) Recommender Systems

B) Auto-completion

C) Microwaves

D) Drug Discovery

Answer in the cell below using the uppercase letter associated with your answer. Place your answer between "", assign the correct answer to an object called ***answer1_1****.*

[3]:
```
answer1_1 = "C"

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
answer1_1
```

[3]:
```
'C'
```

[5]:
```
t.test_1_1(answer1_1)
```

[5]:
```
'Success'
```

Question 1.2 {points: 1}

The following are different types of machine learning examples, categorize the as either "Supervised" or "Unsupervised" in a object of the same name.

To answer the question, assign the letter associated with the correct answer to a variable in the code cell below:

A) Filtering emails as junk or regular mail.

B) Grouping companies customers together on similarity.

C) Predicting a student's MCAT grade given the marks they have received on the practice exams.

D) Speech recognition (inferring the words from an audio signal of a person's voice)

Assign the examples to objects called ***supervised*** *and* ***unsupervised*** *in the code chunk below. Make sure your answer is surrounded by square brackets. If there are more than one answers to this question, separate each number with a comma in the square brackets. For example if you believe*

that A and B are supervised, C is unsupervised and D is neither, then your answer would look like this:

```
supervised = ["A", "B"]
unsupervised = ["A"]
```

[7]:
```
supervised = ["A", "C", "D"]
unsupervised = ["B"]

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
[supervised,unsupervised]
```

[7]:
```
[['A', 'C', 'D'], ['B']]
```

[9]:
```
t.test_1_2(supervised,unsupervised)
```

[9]:
```
'Success'
```

Question 1.3 {points: 4}

In the next 3 situations, would the problem be considered ***regression*** or ***classification***?

i) Based on the ingredients in a store, categorizing it as breakfast food, lunch food, dinner food.

ii) Estimating the fiber content in different types of bread based on the ingredients.

iii) The dataset below (with the target column being `number_comments`:

	video_name	number_views	number_likes	number_dislikes	time_since_upload	number_comments
0	Cute kitten purring	345602	3743	89	5603	6
1	How to make gua-camole	56777032	69327	3405	25793	107
2	Pool trick shots	2458602	23743	40622	104802	?
3	Light show	103	3	1	300	?
4	Edward Snowden	603042	1502	62	315360	?

Save your answers as a string in an object named as `answer1_3_a`*,* `answer1_3_b` *and* `answer1_3_c` *respectively.*
Example:

```
answer1_3_a = "Regression"
```

[11]:
```
answer1_3_a = "Classification"
answer1_3_b = "Regression"
answer1_3_c = "Regression"

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
```

[13]:
```
# check that the variable exists
assert 'answer1_3_a' in globals(
), "Please make sure that your solution is named 'answer1_3_a'"

# This test has been intentionally hidden. It will be up to you to decide if␣
 ↪your solution
# is sufficiently good.
```

[15]:
```
t.test_1_3_2(answer1_3_b)
```

[15]:
```
'Success'
```

[17]:
```
t.test_1_3_3(answer1_3_c)
```

[17]:
```
'Success'
```

1.3 2. Terminology

Question 2.1 {points: 5}

For this next section, as per the lecture notes, give a synonym for each term.

Example:

`observations = "examples"`

Save your answers as a string in the respective object name aka ***`rows`****,* ***`columns`****,* ***`target`*** *and* ***`training`*** *respectively.*

[29]:
```
rows = "examples"
inputs = "features"
outputs = "target"
training = "fitting"

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
```

[31]:
```
t.test_2_1_1(rows)
```

[31]:
```
'Success'
```

[33]:
```
t.test_2_1_2(inputs)
```

[33]:
```
'Success'
```

[39]:
```
# check that the variable exists
assert 'outputs' in globals(
), "Please make sure that your solution is named 'outputs'"

# This test has been intentionally hidden. It will be up to you to decide if␣
 ↪your solution
# is sufficiently good.
```

[41]:
```
t.test_2_1_4(training)
```

[41]:
```
'Success'
```

Question 2.2

Below is a toy data set that we will be using to get familiar with scikit-learn.

[ ]:
```
toy_data = {
    # Features
    'is_sweet': [0, 0, 1, 1, 0, 1, 0],
    'diameter': [3, 3, 1, 1, 3, 1, 4],
    # Target
    'target': ['Apple', 'Apple', 'Grape', 'Grape', 'Lemon', 'Grape', 'Apple']
}

df = pd.DataFrame(toy_data)
df
```

Question 2.2(a) {points: 1}

How many features are present in the dataset?

Assign your answer to a variable of type ***int*** *called* ***answer2_2_a***

[43]:
```
answer2_2_a = 2

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
answer2_2_a
```

[43]:
```
2
```

[45]:
```
t.test_2_2a(answer2_2_a)
```

[45]:
```
'Success'
```

Question 2.2(b) {points: 1}

Express the names of the features as strings in a list. ex `["Feature1", "Feature2", "Feature3"]`

Assign your answer to a variable called ***answer2_2_b****.*

[47]:
```
answer2_2_b = ["is_sweet", "diameter"]


# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
answer2_2_b
```

[47]:
```
['is_sweet', 'diameter']
```

[49]:
```
t.test_2_2b(answer2_2_b)
```

[49]:
```
'Success'
```

Question 2.2(c) {points: 1}

How many different target classes (categories) are present in the dataset? *Assign your answer to a variable of type* ***int*** *called* ***answer2_2_c****.*

[53]:
```
answer2_2_c = 3

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
answer2_2_c
```

[53]:
```
3
```

[55]:
```
t.test_2_2c(answer2_2_c)
```

[55]:
```
'Success'
```

2 3. More Terminology

For the next 3 questions we will be working with data that was obtained from The UCI Machine Learning Repository that examines the fertility among males between the ages of 18-36. The data was provided by 100 volunteers who provided a sample and was analyzed according to the WHO 2010 criteria. We have modified this dataset so that it is easier to work with in this assignment.

The columns in the dataset can be explained as follows:

- `age`: Age at the time of analysis. 18-36
- `diseases`: Has the respondent had any childhood diseases (ie , chicken pox, measles, mumps, polio) (No: 0, Yes: 1)(0, 1)
- `accident`: Has the respondent had an accident or serious trauma? (No: 0, Yes: 1)
- `surgical_inter`: Has the respondent had any surgical intervention (No: 0, Yes: 1)
- `fever`: Has the respondent had high fevers in the last year? (Less than three months ago: -1, More than three months ago:0, No: 1)

- `alcohol`: What is the respondent's frequency of alcohol consumption? (Several times a day: 0.2, Every day: 0.4, Several times a week: 0.6, Once a week: 0.8, Hardly ever or never: 1)
- `smoking_level`: Smoking habit 1) never, 2) occasional 3) daily (-1, 0, 1)
- `sitting_time`: Number of hours spent sitting per day (Minumum value:0, Maximum value: 1)
- `diagnosis`: Output: Diagnosis normal (N), altered (O)

The column `diagnosis` is our target.

[57]:
```
fertility_df = pd.read_csv('data/fertility_diagnosis.csv')
fertility_df.head()
```

[57]:

	age	diseases	accident	surgical_inter	fever	alcohol	smoking_level	\
0	25.0	0	1	1	0	0.8	0	
1	34.0	1	0	1	0	0.8	1	
2	18.0	1	0	0	0	1.0	-1	
3	27.0	0	1	1	0	1.0	-1	
4	24.0	1	1	0	0	0.8	-1	

	sitting_time	diagnosis
0	0.88	N
1	0.31	O
2	0.50	N
3	0.38	N
4	0.50	O

Question 3.1 {points: 1}

How many features are there?

Assign the correct answer to an object of type ***`int`*** *called* ***`feature_num`****.*

[69]:
```
feature_num = 8

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
feature_num
```

[69]: 8

[71]:
```
t.test_3_1(feature_num)
```

[71]: 'Success'

Question 3.2 {points: 2}

What type of problem is this ***Regression*** or ***Classification***?

Answer in the cell below with your answer. Place your answer between "", assign the correct answer to an object called ***`problem`****.*

[77]:
```
problem = "Classification"

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
problem
```

[77]:
```
'Classification'
```

[79]:
```
# check that the variable exists
assert 'problem' in globals(
), "Please make sure that your solution is named 'problem'"

# This test has been intentionally hidden. It will be up to you to decide if␣
 ↪your solution
# is sufficiently good.
```

Question 3.3 {points: 1}

Create your X and y dataframes based on the answers to the questions above.

Save each in the respective object names `X` *and* `y`

[83]:
```
X = fertility_df.drop(columns=["diagnosis"])
y = fertility_df["diagnosis"]

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
```

[85]:
```
t.test_3_3(X,y)
```

[85]:
```
'Success'
```

3 4. Dummy Classifier

For this question we are going to explore the accuracy of different dummy classifiers.

Question 4.1 {points: 0}

Import the `DummyClassifier()` from the `from sklearn.dummy` library.

[87]:
```
# your code here
from sklearn.dummy import DummyClassifier
# raise NotImplementedError # No Answer - remove if you provide an answer
```

[89]:
```
t.test_4_1()
```

[89]:
```
'Success'
```

Question 4.2 {points: 1}

Build a Dummy Classifier with the `strategy` "stratified" and name it `strat_clf`.
Make sure to specify a second argument `random_state=1`. This just makes sure every student will produce the same model and remove and randomness.

The *stratified* `strategy` argument value generates predictions by respecting the training set's class distribution. This means that if a target category only exists 5 times out of 100 examples, the dummy classifier will only predict the category 5% of the time.

[91]:
```
strat_clf = None

# your code here
from sklearn.dummy import DummyClassifier

strat_clf = DummyClassifier(strategy="stratified", random_state=1)

# raise NotImplementedError # No Answer - remove if you provide an answer
```

[93]:
```
t.test_4_2(strat_clf)
```

[93]:
```
'Success'
```

Question 4.3 {points: 1}

Train `strat_clf` on the `X` and `y` objects we made in question 4.2.

Predict on `y` and save this in an object named `strat_predicted`.

[95]:
```
strat_predicted = None

# your code here
# Fit the model
strat_clf.fit(X, y)

# Predict using the model
strat_predicted = strat_clf.predict(X)

# raise NotImplementedError # No Answer - remove if you provide an answer
```

[97]:
```
t.test_4_3(strat_clf,strat_predicted)
```

[97]:
```
'Success'
```

Question 4.4 {points: 2}

What is the accuracy of this model?

Save the result to 2 decimal places in an object named `strat_accuracy`.

[99]:
```
from sklearn.metrics import accuracy_score
strat_accuracy = round(accuracy_score(y, strat_predicted), 2)
```

```
# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer

strat_accuracy
```

[99]: 0.77

[101]:
```
# check that the variable exists
assert 'strat_accuracy' in globals(
), "Please make sure that your solution is named 'strat_accuracy'"

# This test has been intentionally hidden. It will be up to you to decide if␣
 ↪your solution
# is sufficiently good.
```

Question 4.5 {points: 1}

We can train different models and compare which ones are the most accurate using a `for` loop.

For this question make a `for` loop that loops over the values "stratified", "most_frequent" and "uniform", builds a model, trains and scores each model and save the accuracy to 2 decimal places in a new list called `accuracies`.

Make sure to specify a second argument `random_state=1` for the models.

[103]:
```
strategies = ["stratified", "most_frequent", "uniform"]
accuracies = list()

# your code here
for strategy in strategies:
    # Create and train the DummyClassifier
    dummy_clf = DummyClassifier(strategy=strategy, random_state=1)
    dummy_clf.fit(X, y)

    # Predict and calculate accuracy
    y_pred = dummy_clf.predict(X)
    acc = round(accuracy_score(y, y_pred), 2)

    # Append accuracy to the list
    accuracies.append(acc)
# raise NotImplementedError # No Answer - remove if you provide an answer
set(accuracies)
```

[103]: {0.45, 0.77, 0.88}

[105]:
```
t.test_4_5(accuracies)
```

[105]: 'Success'

Question 4.6 {points: 1}

Create a new dataframe name `accuracies_df` made from the lists `strategies` and `accuracies`. It should look like this:

	strategy	accuracy
0	stratified	#
1	most_frequent	#
2	uniform	#

[109]:
```
import pandas as pd

accuracies_df = pd.DataFrame({
    "strategy": strategies,
    "accuracy": accuracies
})

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
accuracies_df
```

[109]:

	strategy	accuracy
0	stratified	0.77
1	most_frequent	0.88
2	uniform	0.45

[111]:
```
t.test_4_6(accuracies_df)
```

[111]:
```
'Success'
```

Question 4.7 {points: 1}

Using Altair, make a bar plot (`mark_bar()`) that displays the `strategy` on the x-axis and the `accuracy` on the y-axis. Don't forget to give it a title and to sort it in ascending order. Make sure it has the dimensions `width=500, height=300`. Name the plot `Dummy_plot`.

[113]:
```
import altair as alt

Dummy_plot = alt.Chart(accuracies_df).mark_bar().encode(
    x=alt.X("strategy", sort="ascending", title="Strategy"),
    y=alt.Y("accuracy", title="Accuracy"),
).properties(
    title="Accuracy by Strategy",
    width=500,
    height=300
)

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
Dummy_plot
```

```
C:\Users\diksh\anaconda3\Lib\site-packages\altair\utils\core.py:395:
FutureWarning: the convert_dtype parameter is deprecated and will be removed in
a future version.  Do ``ser.astype(object).apply()`` instead if you want
``convert_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

[113]:
```
alt.Chart(…)
```

[115]:
```
t.test_4_7(Dummy_plot)
```

[115]:
```
'Success'
```

Question 4.8 {points: 1}

Which strategy had the best results?

To answer the question, assign the letter associated with the correct answer to a variable in the code cell below:

A) `stratified`

B) `most_frequent`

C) `uniform`

Answer in the cell below using the uppercase letter associated with your answer. Place your answer between "", assign the correct answer to an object called ***answer4_8****.*

[121]:
```
answer4_8 = "most_frequent"

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
answer4_8
```

[121]:
```
'most_frequent'
```

[123]:
```
t.test_4_8(answer4_8)
```

[123]:
```
'Success'
```

4 5. Dummy Regressor

This time instead of focusing on a classification problem we are going to focus on a regression problem using the `DummyRegressor` baseline model.

For this question we are using a dataset obtained from The UCI Machine Learning Repository that contains the market historical data of real estate valuation collected from Sindian District, New Taipei City in Taiwan.

The columns in the dataset can be explained as follows:

- `date`: the transaction date (for example, 2013.250=2013 March, 2013.500=2013 June, etc.)
- `house_age`: the house age (unit: year)

- `distance_station`: the distance to the nearest Mass Rapid Transit (MRT) station (unit: meter)
- `num_stores`: the number of convenience stores in the living circle on foot (integer)(a ***living circle*** is a residential space with similar local characteristics, and daily behaviors)
- `latitude`: the geographic coordinate, latitude. (unit: degree)
- `longitude`: the geographic coordinate, longitude. (unit: degree)
- `price`: house price per unit area (10000 New Taiwan Dollar/Ping,where Ping is a local unit of area, 1 Ping = 3.3 meter squared)

[125]:
```
housing_df = pd.read_csv('data/real_estate.csv')
housing_df.head()
```

[125]:

	house_age	distance_station	num_stores	latitude	longitude	price
0	32.0	84.87882	10	24.98298	121.54024	37.9
1	19.5	306.59470	9	24.98034	121.53951	42.2
2	13.3	561.98450	5	24.98746	121.54391	47.3
3	13.3	561.98450	5	24.98746	121.54391	54.8
4	5.0	390.56840	5	24.97937	121.54245	43.1

Question 5.1 {points: 1}

Create your X and y dataframes.
For the `X` dataframe make sure that you are not including `price`. Since our `y` (target) is the `price` column.

Save each in the respective object names ***`X`*** *and* ***`y`****.*

[127]:
```
# Define X and y
X = housing_df.drop(columns=["price"])  # Features
y = housing_df["price"]  # Target

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
```

[129]:
```
t.test_5_1(X,y)
```

[129]:
```
'Success'
```

Question 5.2 {points: 0}

Import `DummyRegressor()` from the `from sklearn.dummy` library.

[131]:
```
# your code here
from sklearn.dummy import DummyRegressor
# raise NotImplementedError # No Answer - remove if you provide an answer
```

[133]:
```
t.test_5_2()
```

[133]:
```
'Success'
```

Question 5.3 {points: 1}

Build a Dummy Regressor with the `strategy` "mean" and name it `dummy_regr`. Train it on the variables `X` and `y` that we made in question 5.1. Predict on the `y` object and save the output in a object name `reg_predict`. Is the output what you expected?

[135]:
```
# Create the Dummy Regressor
dummy_regr = DummyRegressor(strategy="mean")

# Train the regressor
dummy_regr.fit(X, y)

# Predict on the training data
reg_predict = dummy_regr.predict(X)

# raise NotImplementedError # No Answer - remove if you provide an answer
```

[137]:
```
t.test_5_3(dummy_regr,reg_predict)
```

[137]:
```
'Success'
```

Question 5.4 {points: 1}

To calculate the accuracy of regression models we need to see how close we were to the true value. We can use `.score()` like we did for the Dummy Classifier but this measure something different. The output of `.score()` for regressors returns the coefficient of determination of the prediction R^2. The best possible score is 1.0 and it can be a negative (because the model can be arbitrarily worse).

Since a dummy regressor always predicts the mean value and disregards the input features(`X`), we expect a `.score()` of 0.0.

Try it out on our real estate valuation data below and save the result in an object named `reg_score`.

[139]:
```
# Calculate the R² score
reg_score = dummy_regr.score(X, y)

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
reg_score
```

[139]:
```
0.0
```

[141]:
```
t.test_5_4(reg_score)
```

[141]:
```
'Success'
```

Question 5.5 {points: 1}

For baseline models, the features of the data are not taken into consideration when making predictions. This will change for other models that we will be learning. In fact, let's take a look at the feature `distance_station` which measures the distance a house is from a Mass Rapid Transit station vs the house price in the graph below:

[143]:
```
housing_plot = alt.Chart(housing_df, width=500, height=300).mark_circle(
    opacity=0.7).encode(
    alt.X('price:Q'),
    alt.Y('distance_station:Q'))
housing_plot
```

[143]:
```
alt.Chart(…)
```

Looking at the graph, do you think that the feature `distance_station` will help in the prediction of housing prices if we were to use models that are not baseline?

A) Yes, since there seems to be a positive relationship between `price` and `distance_station`.

B) Yes, since there seems to be a negative relationship between `price` and `distance_station`.

C) No, since there doesn't seem to be a relationship between the feature and the target.

D) No, since there seems to be a negative relationship between the feature `distance_station` and the target `price`

Answer in the cell below using the uppercase letter associated with your answer. Place your answer between "", assign the correct answer to an object called ***answer5_5****.*

[147]:
```
answer5_5 = "B"

# your code here
# raise NotImplementedError # No Answer - remove if you provide an answer
answer5_5
```

[147]:
```
'B'
```

[149]:
```
t.test_5_5(answer5_5)
```

[149]:
```
'Success'
```

4.1 Before Submitting

Before submitting your assignment please do the following:

- Read through your solutions
- **Go to the file and download ipynb, file –> save and export notebook as pdf, submit both ipynb and pdf file**
- Makes sure that none of your code is broken
- Verify that the tests from the questions you answered have obtained the output "Success"

This is a simple way to make sure that you are submitting all the variables needed to mark the assignment. This method should help avoid losing marks due to changes in your environment.

4.2 Attributions

- Fertitily Diagnosis Dataset: - The UCI Machine Learning Repository

David Gil, Jose Luis Girela, Joaquin De Juan, M. Jose Gomez-Torres, and Magnus Johnsson. Predicting seminal quality with artificial intelligence methods. Expert Systems with Applications, 39(16):12564 â€“ 12573, 2012

- Real Estate Dataset - The UCI Machine Learning Repository

[ ]: