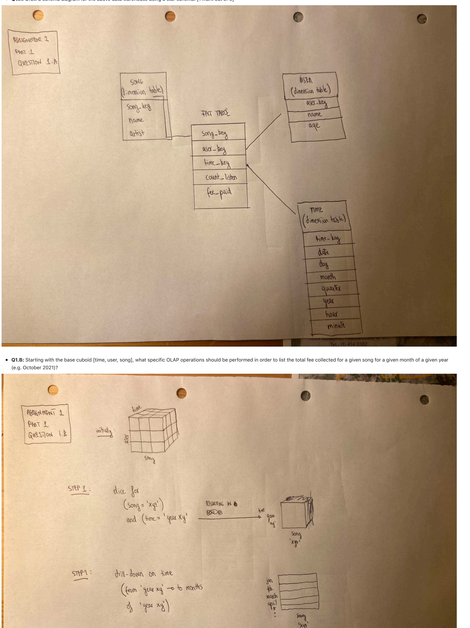
Assignment 2

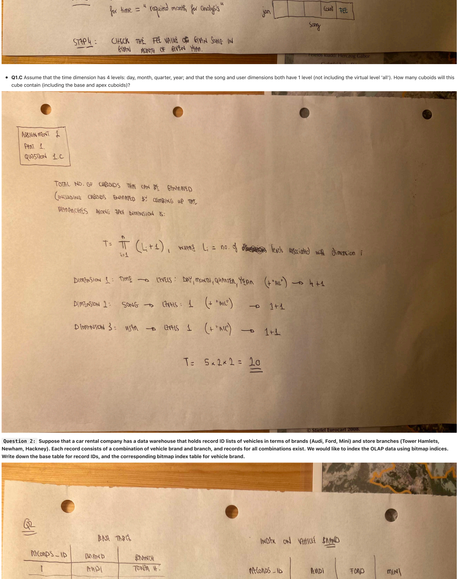
Part 1

Question 1: A data warehouse for a music streaming company consists of the dimensions song, user, time (time and date of when the user listened to a song), and the two measures count (how many times a user listened to the song) and fee (fee paid by the streaming company to the artist every time a user listens to that song).

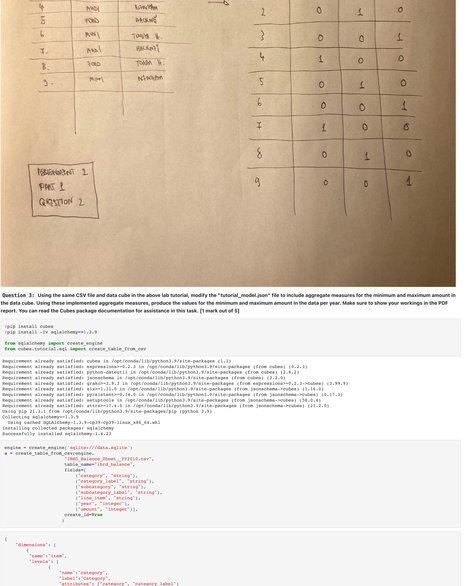
- **Q1.A**: Draw a schema diagram for the above data warehouse using a star schema. [1 mark out of 5]



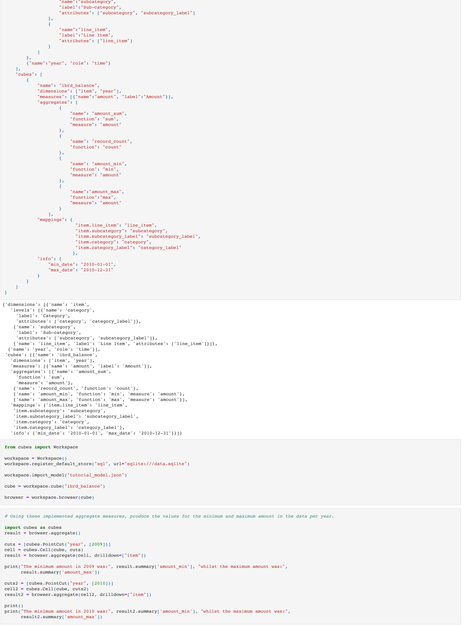
- **Q1.B**: Starting with the base cuboid [time, user, song], what specific OLAP operations should be performed in order to list the total fee collected for a given song for a given month of a given year (e.g. October 2021)?



- **Q1.C**: Assume that the time dimension has 4 levels: day, month, quarter, year, and that the song and user dimensions both have 1 level (not including the virtual level 'all'). How many cuboids will this cube contain (including the base and apex cuboids)?



Question 2: Suppose that a car rental company has a data warehouse that holds record ID lists of vehicles in terms of brands (Audi, Ford, Mini) and store branches (Tower Hamlets, Newham, Hackney). Each record consists of a combination of vehicle brand and branch, and records for all combinations exist. We would like to index the OLAP data using bitmap indices. Write down the base table for record IDs, and the corresponding bitmap index table for vehicle brand.



Question 3: Using the same CSV file and data cube in the above lab tutorial, modify the "tutorial_model.json" file to include aggregate measures for the minimum and maximum amount in the data cube. Using these implemented aggregate measures, produce the values for the minimum and maximum amount in the data per year. Make sure to show your workings in the PDF report. You can read the Cubes package documentation for assistance in this task. [1 mark out of 5]

```
In [1]:
!pip install cubes
!pip install -Iv sqlalchemy==1.3.9

from sqlalchemy import create_engine
from cubes.tutorial.sql import create_table_from_csv
```

```
Requirement already satisfied: cubes in /opt/conda/lib/python3.9/site-packages (1.1)
Requirement already satisfied: expressions>=0.2.1 in /opt/conda/lib/python3.9/site-packages (from cubes) (0.2.3)
Requirement already satisfied: python-dateutil in /opt/conda/lib/python3.9/site-packages (from cubes) (2.8.2)
Requirement already satisfied: jsonschema in /opt/conda/lib/python3.9/site-packages (from cubes) (3.2.0)
Requirement already satisfied: grako>=3.9.2 in /opt/conda/lib/python3.9/site-packages (from expressions>=0.2.1->cubes) (3.99.9)
Requirement already satisfied: six>=1.11.0 in /opt/conda/lib/python3.9/site-packages (from jsonschema->cubes) (1.16.0)
Requirement already satisfied: pyrsistent>=0.14.0 in /opt/conda/lib/python3.9/site-packages (from jsonschema->cubes) (0.17.3)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.9/site-packages (from jsonschema->cubes) (58.0.4)
Requirement already satisfied: attrs>=17.4.0 in /opt/conda/lib/python3.9/site-packages (from jsonschema->cubes) (21.2.0)
Using pip 21.3.1 from /opt/conda/lib/python3.9/site-packages/pip (python 3.9)
Collecting sqlalchemy==1.3.9
  Using cached SQLAlchemy-1.3.9-cp39-cp39-linux_x86_64.whl
Installing collected packages: sqlalchemy
Successfully installed sqlalchemy-1.3.23
```

```
In [2]:
engine = create_engine('sqlite:///data.sqlite')
s = create_table_from_csv(engine,
                          "New_Sheet__FY2510.csv",
                          table_name="ibrd_balance",
                          fields=[
                              ("category", "string"),
                              ("category_label", "string"),
                              ("subcategory", "string"),
                              ("subcategory_label", "string"),
                              ("line_item", "string"),
                              ("year", "integer"),
                              ("amount", "integer")],
                          create_id=True
                          )
```

```
In [3]:
{
    "dimensions": [
        {
            "name":"item",
            "levels": [
                {
                    "name":"category",
                    "label":"Category",
                    "attributes": ["category", "category_label"]
                },
                {
                    "name":"subcategory",
                    "label":"Sub-category",
                    "attributes": ["subcategory", "subcategory_label"]
                },
                {
                    "name":"line_item",
                    "label":"Line item",
                    "attributes": ["line_item"]
                }
            ]
        },
        {"name":"year", "role": "time"}
    ],
    "cubes": [
        {
            "name": "ibrd_balance",
            "dimensions": ["item", "year"],
            "measures": [{"name":"amount", "label":"Amount"}],
            "aggregates": [
                {
                    "name": "amount_sum",
                    "function": "sum",
                    "measure": "amount"
                },
                {
                    "name": "record_count",
                    "function": "count"
                },
                {
                    "name": "amount_min",
                    "function": "min",
                    "measure": "amount"
                },
                {
                    "name":"amount_max",
                    "function":"max",
                    "measure": "amount"
                }
            ],
            "mappings": {
                            "item.line_item": "line_item",
                            "item.subcategory": "subcategory",
                            "item.subcategory_label": "subcategory_label",
                            "item.category": "category",
                            "item.category_label": "category_label"
                        },
            "info": {
                    "min_date": "2010-01-01",
                    "max_date": "2010-12-31"
                }
        }
    ]
}
```

```
Out[3]:
{'dimensions': [{'name': 'item',
   'levels': [{'name': 'category',
     'label': 'Category',
     'attributes': ['category', 'category_label']},
    {'name': 'subcategory',
     'label': 'Sub-category',
     'attributes': ['subcategory', 'subcategory_label']},
    {'name': 'line_item', 'label': 'Line item', 'attributes': ['line_item']}]},
  {'name': 'year', 'role': 'time'}],
 'cubes': [{'name': 'ibrd_balance',
   'dimensions': ['item', 'year'],
   'measures': [{'name': 'amount', 'label': 'Amount'}],
   'aggregates': [{'name': 'amount_sum',
     'function': 'sum',
     'measure': 'amount'},
    {'name': 'record_count', 'function': 'count'},
    {'name': 'amount_min', 'function': 'min', 'measure': 'amount'},
    {'name': 'amount_max', 'function': 'max', 'measure': 'amount'}],
   'mappings': {'item.line_item': 'line_item',
    'item.subcategory': 'subcategory',
    'item.subcategory_label': 'subcategory_label',
    'item.category': 'category',
    'item.category_label': 'category_label'},
   'info': {'min_date': '2010-01-01', 'max_date': '2010-12-31'}}]}
```

```
In [4]:
from cubes import Workspace

workspace = Workspace()
workspace.register_default_store("sql", url="sqlite:///data.sqlite")

workspace.import_model("tutorial_model.json")

cube = workspace.cube('ibrd_balance')

browser = workspace.browser(cube)
```

```
In [5]:
# Using these implemented aggregate measures, produce the values for the minimum and maximum amount in the data per year.

import cubes as cubes
result = browser.aggregate()

cut4 = cubes.PointCut("year", [2009])
cell = cubes.Cell(cube, cut4)
result = browser.aggregate(cell, drilldown="item")

print("The minimum amount in 2009 was", result.summary['amount_min'], "whilst the maximum amount was",
      result.summary['amount_max'])

cut2 = cubes.PointCut("year", [2010])
cell2 = cubes.Cell(cube, cut2)
result2 = browser.aggregate(cell2, drilldown="item")

print()
print("The minimum amount in 2010 was", result2.summary['amount_min'], "whilst the maximum amount was",
      result2.summary['amount_max'])
```

```
The minimum amount in 2009 was -1683 whilst the maximum amount was: 110040

The minimum amount in 2010 was -3043 whilst the maximum amount was: 128577
```

In [1]:
```
!pip install cubes
!pip install -Iv sqlalchemy==1.3.9
```

```
Requirement already satisfied: cubes in /opt/conda/lib/python3.9/site-packages (1.1)
Requirement already satisfied: python-dateutil in /opt/conda/lib/python3.9/site-packages (from cubes) (2.8.2)
Requirement already satisfied: jsonschema in /opt/conda/lib/python3.9/site-packages (from cubes) (3.2.0)
Requirement already satisfied: expressions>=0.2.3 in /opt/conda/lib/python3.9/site-packages (from cubes) (0.2.3)
Requirement already satisfied: grako>=3.9.3 in /opt/conda/lib/python3.9/site-packages (from expressions>=0.2.3->cubes) (3.99.9)
Requirement already satisfied: attrs>=17.4.0 in /opt/conda/lib/python3.9/site-packages (from jsonschema->cubes) (21.2.0)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.9/site-packages (from jsonschema->cubes) (58.0.4)
Requirement already satisfied: pyrsistent>=0.14.0 in /opt/conda/lib/python3.9/site-packages (from jsonschema->cubes) (0.17.3)
Requirement already satisfied: six>=1.11.0 in /opt/conda/lib/python3.9/site-packages (from jsonschema->cubes) (1.16.0)
Using pip 21.3.1 from /opt/conda/lib/python3.9/site-packages/pip (python 3.9)
Collecting sqlalchemy==1.3.9
  Using cached SQLAlchemy-1.3.9-cp39-cp39-linux_x86_64.whl
Installing collected packages: sqlalchemy
Successfully installed sqlalchemy-1.4.23
```

`Question 4:` **Using the CSV file "country-income.csv" (found in the week 5 supplementary lab documents), perform the following:**

- **Q4.A: Load the CSV file using Cubes, create a JSON file for the data cube model, and create a data cube for the data. Use as dimensions the region, age, and online shopper fields. Use as measure the income. Define aggregate functions in the data cube model for the total, average, minimum, and maximum income. In your PDF report, show the relevant scripts and files created.**

In [2]:
```
from sqlalchemy import create_engine
from cubes.tutorial.sql import create_table_from_csv
```

In [3]:
```
# Load the CSV file using Cubes
engine = create_engine('sqlite:///data.sqlite')
```

In [4]:
```
{
    "dimensions": [
        {
         "name":"region",
         "levels": [
                {
                    "name":"region",
                    "label":"Region",
                    "attributes": ["region"]
                }
            ]
        },
        {
          "name": "age",
           "levels":  [
                {
                    "name":"age",
                    "label":"Age",
                    "attributes": ["age"]
                }
            ]
        },
        {
         "name": "online_shopper",
         "levels": [
                {
                    "name":"online_shopper",
                    "label":"Online_Shopper",
                    "attributes": ["online_shopper"]
                }
            ]
        }

             ],

    "cubes": [
        {
            "name": "country_income",
            "dimensions": ["region", "age", "online_shopper"],
            "measures": [{"name":"income", "label":"Income"}],
            "aggregates": [
                    {
                        "name": "income_total",
                        "function": "sum",
                        "measure": "income"
                    },
                    {
                        "name": "income_min",
                        "function": "min",
                        "measure" : "income"
                    },
                    {
                        "name": "income_max",
                        "function": "max",
                        "measure": "income"
                    },
                    {
                        "name": "income_average",
                        "function": "avg",
                        "measure": "income"
                    },
                    {
                        "name": "count",
                        "function": "count",
                        "measure":"income"
                    }
                ]
          }
        ]
    }
```

Out[4]:
```
{'dimensions': [{'name': 'region',
   'levels': [{'name': 'region',
     'label': 'Region',
     'attributes': ['region']}]},
  {'name': 'age',
   'levels': [{'name': 'age', 'label': 'Age', 'attributes': ['age']}]},
  {'name': 'online_shopper',
   'levels': [{'name': 'online_shopper',
     'label': 'Online_Shopper',
     'attributes': ['online_shopper']}]}],
 'cubes': [{'name': 'country_income',
   'dimensions': ['region', 'age', 'online_shopper'],
   'measures': [{'name': 'income', 'label': 'Income'}],
   'aggregates': [{'name': 'income_total',
     'function': 'sum',
     'measure': 'income'},
    {'name': 'income_min', 'function': 'min', 'measure': 'income'},
    {'name': 'income_max', 'function': 'max', 'measure': 'income'},
    {'name': 'income_average', 'function': 'avg', 'measure': 'income'},
    {'name': 'count', 'function': 'count', 'measure': 'income'}]}]}
```

In [5]:
```
create_table_from_csv(engine,
                      "country-income.csv",
                      table_name="country_income",
                      fields=[
                          ("region", "string"),
                          ("age", "integer"),
                          ("income", "integer"),
                          ("online_shopper", "string")],
                      create_id=True
                     )
```

In [6]:
```
from cubes import Workspace

workspace = Workspace()
workspace.register_default_store("sql", url="sqlite:///data.sqlite")
```

In [7]:
```
workspace.import_model('my_code.json')

cube = workspace.cube("country_income")
```

- **Q4.B: Using the created data cube and data cube model, produce aggregate results for: the whole data cube; results per region; results per online shopping activity; and results for all people aged between 40 and 50.**

In [8]:
```
# Aggregate result for whole data cube
browser = workspace.browser(cube)
result = browser.aggregate()
print("Aggregate results for the whole data cube:", result.summary)

print()

# Aggregate results per region
result = browser.aggregate(drilldown=["region"])
print("Aggregate results per region:")
for record in result:
    print(record)

print()

# Aggregate results per online shopping activity
result = browser.aggregate(drilldown=["online_shopper"])
print("Aggregate results per online shopping activity:")
for record in result:
    print(record)

print()

import cubes as cubes
# Aggregate results for all people aged between 40 and 50
cuts = [cubes.RangeCut("age", ["40"], ["50"])]
cell = cubes.Cell(cube, cuts)
result = browser.aggregate(cell)
print("Aggregate results for all people aged between 40 and 50:", result.summary)
```

```
Aggregate results for the whole data cube: {'income_total': 768200, 'income_min': 57600, 'income_max': 99600, 'income_average': 76820.0, 'count': 10}

Aggregate results per region:
{'region': 'Brazil', 'income_total': 193200, 'income_min': 57600, 'income_max': 73200, 'income_average': 64400.0, 'count': 3}
{'region': 'India', 'income_total': 331200, 'income_min': 69600, 'income_max': 94800, 'income_average': 82800.0, 'count': 4}
{'region': 'USA', 'income_total': 243800, 'income_min': 64800, 'income_max': 99600, 'income_average': 81266.66666666667, 'count': 3}

Aggregate results per online shopping activity:
{'online_shopper': 'No', 'income_total': 386400, 'income_min': 62400, 'income_max': 99600, 'income_average': 77280.0, 'count': 5}
{'online_shopper': 'Yes', 'income_total': 381800, 'income_min': 57600, 'income_max': 94800, 'income_average': 76360.0, 'count': 5}

Aggregate results for all people aged between 40 and 50: {'income_total': 451400, 'income_min': 62400, 'income_max': 86400, 'income_average': 75233.33333333333, 'count'
: 6}
```

Assignment 2

Part 2

`Question 1:` **Consider a dataset $\mathcal{D}$ that contains only two observations $\mathbf{x}_1 = (1, -1)$ and $\mathbf{x}_2 = (-1, 1)$. Suppose that the class of the first observation is $y_1 = 0$ and that the class of the second observation is $y_2 = 1$. How would a 1-nearest neighbour classifier based on the Euclidean distance classify the observation $\mathbf{x} = (-2, 3)$? What are the distances between this new observation and each observation in the dataset?**

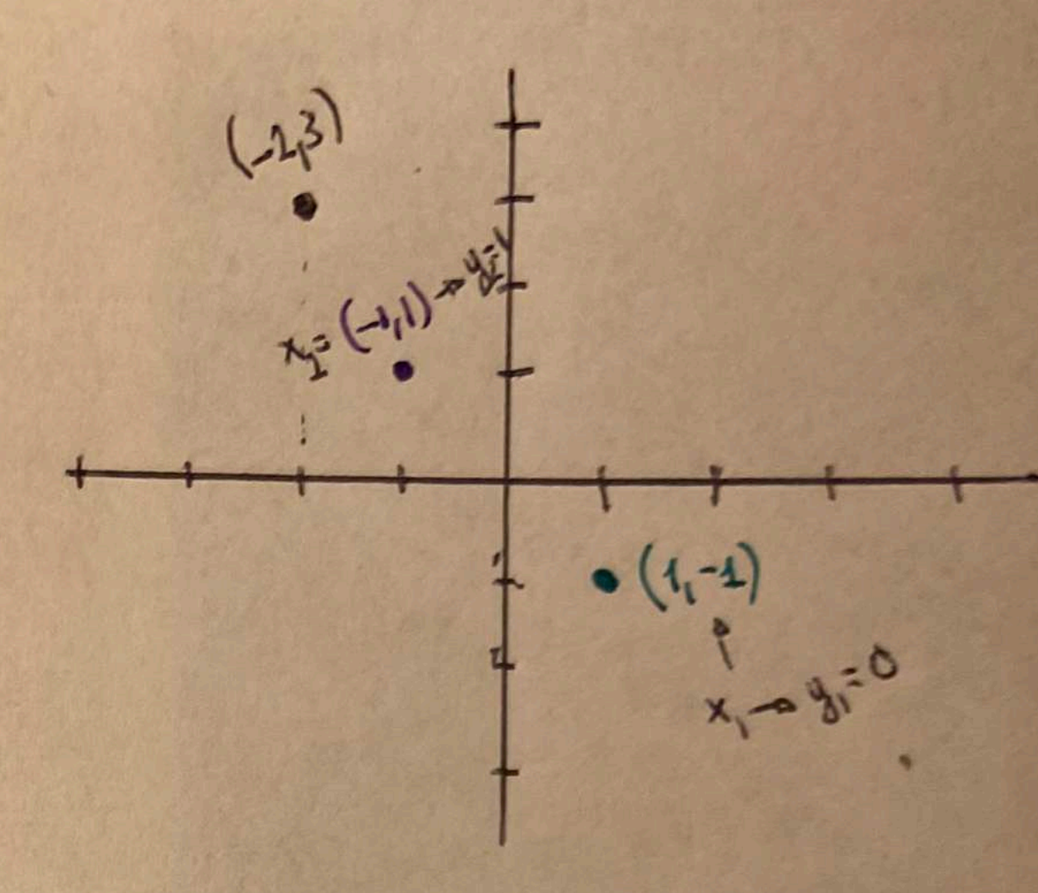
1. CONSIDER A DATASET D THAT CONTAINS ONLY TWO OBSERVATIONS $X_1 = (1, -1)$ AND $X_2 = (-1, 1)$. SUPPOSE THAT THE CLASS OF THE FIRST OBSERVATION IS $y_1 = 0$ AND THAT THE CLASS OF THE SECOND OBSERVATION IS $y_2 = 1$. HOW WOULD A 1-NEAREST NEIGHBOUR CLASSIFIER BASED ON THE EUCLIDEAN DISTANCE CLASSIFY THE OBSERVATION $X = (-2, 3)$? WHAT ARE THE DISTANCES BTW. THIS NEW OBSERVATION AND EACH OBSERVATION IN THE DATASET?

$D = (x_1, y_1), \dots, (x_2, y_2)$

$x_1 = (1, -1) \rightarrow y_1 = 0$

$x_2 = (-1, 1) \rightarrow y_2 = 1$

$x_3 = (-2, 3) \rightarrow y_3 = ?$



EUCLIDEAN DISTANCES

$[x_3 \leftrightarrow x_2]$ $\quad e_1 = \sqrt{(-2+1)^2 + (3-1)^2} = 2.2360$

$[x_3 \leftrightarrow x_1]$ $\quad e_2 = \sqrt{(-2-1)^2 + (3+1)^2} = 5$

↓

SINCE $e_1 < e_2$,

x_3 IS CLOSER TO x_2 (BASED ON EUCLIDEAN DISTANCE), HENCE x_3 IS THEN CONSIDERED AS

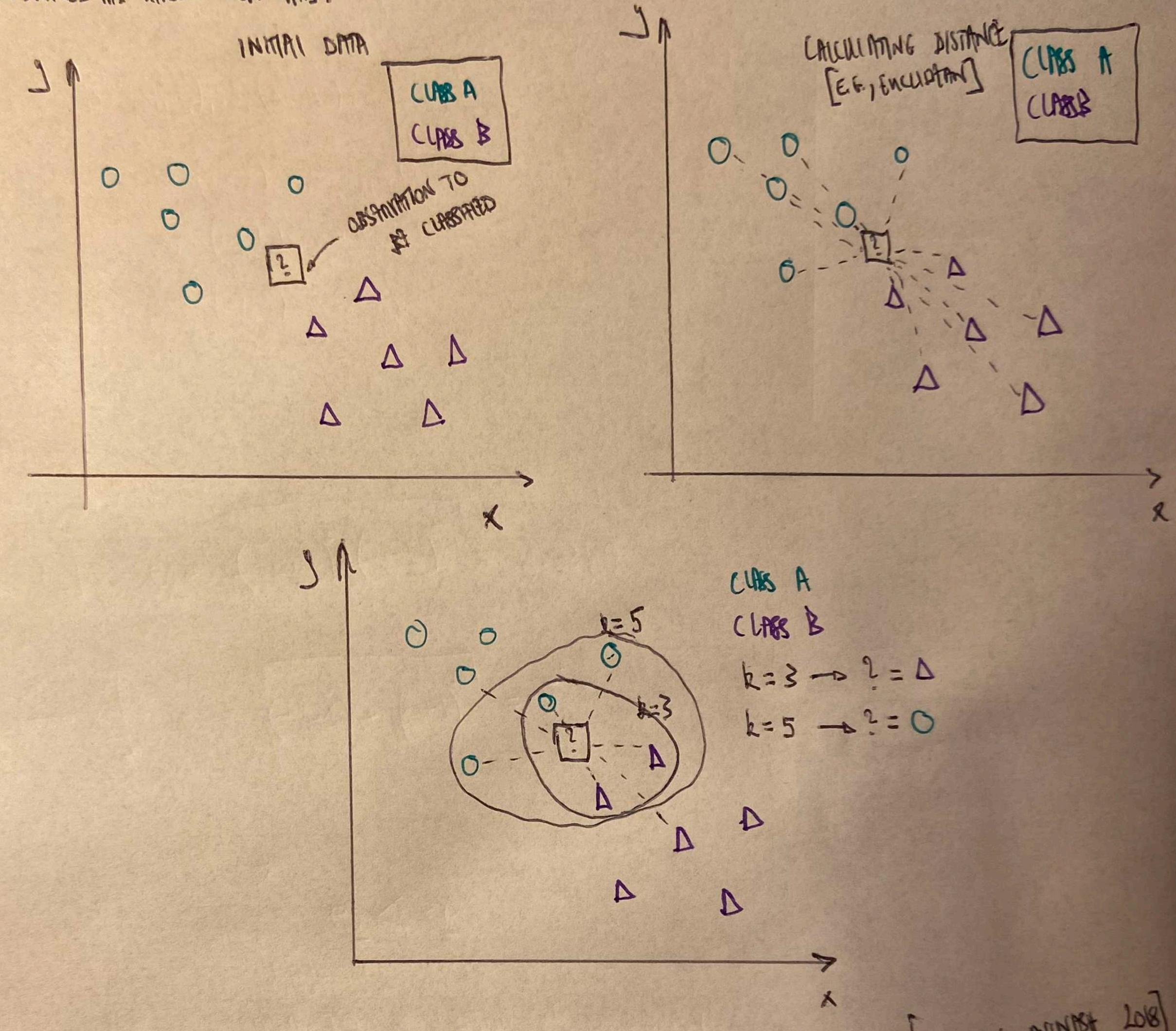
$y_3 = 1$ (BECAUSE $x_2 \Rightarrow y_2 = 1$)

ASSIGNMENT 2
PART 2
QUESTION 1

`Question 2:` **Consider a dataset $\mathcal{D}$ that only contains observations of two different classes. Explain why a k-nearest neighbour classifier does not need a tie-breaking policy when k is odd.**

Q2 CONSIDER A DATASET D THAT ONLY CONTAINS OBSERVATIONS OF TWO DIFFERENT CLASSES. EXPLAIN WHY A k-NEAREST NEIGHBOUR CLASSIFIER DOES NOT NEED A TIE-BREAKING POLICY WHEN k IS ODD

IT'S BEST TO VISUALISE THE REASON FOR THIS.



[NAVLANI, AVINASH 2018]

BECAUSE THE NUMBER OF NEIGHBOURS ON WHOM THE NEW OBSERVATION'S CLASSIFICATION IS BASED IS ODD, THERE WILL ALWAYS BE A "MAJORITY" OF A CLASS (ASSUMING THERE ARE ONLY TWO CLASSES) IN THE SELECTED (ODD) NUMBER OF NEIGHBOURS TO EASILY DECIDE THE ASSIGNED CLASS OF THE NEW OBSERVATION.

ASSIGNMENT 2
PART 2
QUESTION 2

`Question 3:` **Explain why a classifier that obtains an accuracy of 99.9% can be terrible for some datasets.**

ASSIGNMENT 2, PART 2
Q3

CLASSIFICATION ACCURACY = $\dfrac{\text{no. of correct predictions}}{\text{total no. of predictions}}$

THE INTUITION THAT A HIGH CLASSIFICATION ACCURACY MODEL IS SUPERIOR TO A LOWER ACCURACY MODEL AND THAT A HIGH ACCURACY MODEL CLASSIFIER WILL DO WELL ON DATASETS (ALWAYS) BREAKS DOWN WHEN THE DISTRIBUTION OF OBSERVATIONS TO CLASSES ARE IMBALANCED/SKEWED. FOR THAT REASON, GAINING A 99.9% ACCURACY CLASSIFIER ON A BALANCED DATASET DOES NOT MEAN IT WILL DO WELL ON AN IMBALANCED DATASET. ON THE CONTRARY, IT CAN BE EXTREMELY MISLEADING AND DANGEROUS!

- FOR EXAMPLE, IF THE OCCURRENCE OF CLASS "A" IS DOMINATING OVER THE OTHER CLASS (2 CLASSES IN THIS CASE) IN A DATASET, MAYBE BEING FOUND 99% OF THE CASES, THEN THE CLASSIFIER PREDICTING THAT EVERY CASE IS GOING TO BE CLASS "A" WILL HAVE AN ACCURACY OF 99%.

↑ WITHOUT CONSIDERING ANY OTHER ALTERNATIVE

WIKIPEDIA'S EXAMPLE:

A CITY OF 1 MILLION PEOPLE HAS 10 TERRORISTS.

ACTUAL CLASS \ PREDICTED CLASS	TERRORIST	NOT TERRORIST	SUM
TERRORIST	10 TP	0 FN	10
NOT TERRORIST	990 FP	999,000 TN	999,990
SUM	1000	999,000	1,000,000

TP = TRUE POSITIVE
FP = FALSE POSITIVE
FN = FALSE NEGATIVE
TN = TRUE NEGATIVE

ACCURACY = $\dfrac{TP + TN}{(TP + TN + FP + FN)}$ =

$= \dfrac{10 + 999{,}000}{10 + 999000 + 0 + 990} = 0.99901 \sim$ 99.9%

YET, 990 OUT OF THE 1000 POSITIVE PREDICTIONS ARE INCORRECT!!

`Question 4:` **Consider a classifier tasked with predicting whether an observation belongs to class y (positive class). Suppose that this classifier has precision 1.0 and recall 0.1 on a test dataset. If this classifier predicts that an observation does not belong to class y, should it be trusted? Should it be trusted if it predicts that the observation belongs to class y?**

ASSIGNMENT 2, PART 2, Q4

CONSIDER A CLASSIFIER TASKED WITH PREDICTING WHETHER AN OBSERVATION BELONGS TO CLASS y (POSITIVE CLASS). SUPPOSE THAT THIS CLASSIFIER HAS PRECISION 1.0 AND RECALL 0.1 ON A TEST DATASET. IF THIS CLASSIFIER PREDICTS THAT AN OBSERVATION DOES NOT BELONG TO CLASS y, SHOULD IT BE TRUSTED? SHOULD IT BE TRUSTED IF IT PREDICTS THAT THE OBSERVATION BELONGS TO CLASS y?

PRECISION: NUMBER OF TRUE POSITIVES DIVIDED BY THE NUMBER OF TRUE POSITIVES AND FALSE POSITIVES. IT CAN BE THOUGHT OF AS A MEASURE OF A CLASSIFIER'S EXACTNESS. A LOW PRECISION THEN, CAN ALSO MEAN A LARGE NUMBER OF FALSE POSITIVES.

→ $\text{Precision} = \dfrac{N_{TP}}{N_{TP} + N_{FP}}$

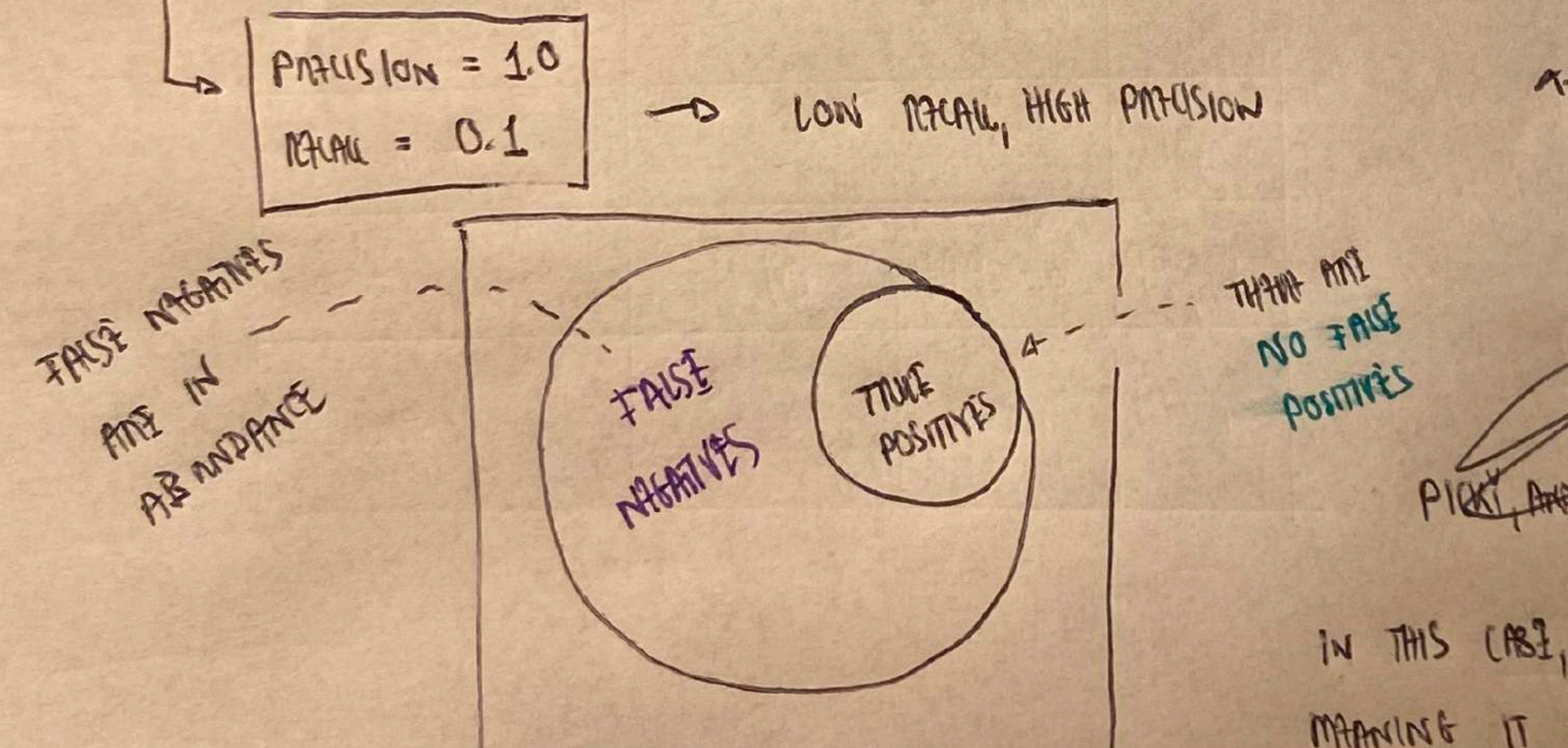
- INTERPRETATION: PERCENTAGE OF POSITIVE PREDICTIONS THAT WERE CORRECT.
- HIGH PRECISION: HIGH CONFIDENCE IN POSITIVE PREDICTIONS.
- PERFECT PRECISION: NO MISTAKES IN POSITIVE PREDICTIONS

RECALL: NUMBER OF TRUE POSITIVES DIVIDED BY THE NUMBER OF TRUE POSITIVES AND THE NUMBER OF FALSE NEGATIVES. IT CAN BE THOUGHT OF AS A MEASURE OF CLASSIFIERS' COMPLETENESS. A LOW RECALL INDICATES MANY FALSE NEGATIVES.

→ $\text{Recall} = \dfrac{N_{TP}}{N_{TP} + N_{FN}}$

(KLINTBERG, ANDREAS: 2017)

PRECISION = 1.0
RECALL = 0.1
→ LOW RECALL, HIGH PRECISION



~~IN THIS CASE THE CLASSIFIER IS VERY PICKY AND "THINKS"~~

IN THIS CASE, THE CLASSIFIER IS QUITE 'PICKY', MEANING IT DOESN'T THINK MANY OBSERVATIONS BELONG TO CLASS 'y'. HOWEVER, WHEN IT SAYS THAT OBS. BELONGS TO CLASS 'y', THEN THOSE OBSERVATIONS ARE TRULY CLASS 'y' (HENCE PRECISION IS MAXIMUM). ON THE OTHER HAND, WHEN IT DECIDES THAT OBSERVATION DOES NOT BELONG TO CLASS 'y', IT SHOULD NOT BE TRUSTED BECAUSE OF THE MANY FALSE NEGATIVES IT PRODUCES (HENCE RECALL IS EXTREMELY LOW)

`Question 5:` **Based on the confusion matrix shown in this lab notebook, what is the pair of classes that is most confusing for the 1-nearest neighbour classifier trained in the previous sections?**

ASSIGNMENT 2
PART 2
QUESTION 5

CLASS 4 AND 9 ARE THE MOST CONFUSING FOR THE 1-NEAREST NEIGHBOUR CLASSIFIER AS CLASS 4 WAS PREDICTED 4 TIMES (WRONGLY) AS CLASS 9.

[DIAGONAL → OBSERVATIONS CLASSIFIED CORRECTLY,
COLUMNS → TRUE LABELS,
ROWS → PREDICTED LABELS
EVERYTHING ABOVE/BELOW DIAGONAL → OBSERVATIONS MISCLASSIFIED!]

`Question 6:` **Train a support vector machine classifier using the same (subsampled) training dataset used in the previous sections and compute its accuracy on the corresponding test dataset. You can use the default hyperparameters for the class `SVC` from `sklearn.svm`. Show the code in the report.**

In [1]:

```python
%config InlineBackend.figure_formats = set(['retina'])
# Importing necessary libraries
import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')

# Selecting the training data from the original dataset
file = open("mnist.pkl",'rb')
X, y = pickle.load(file, encoding='latin1')[0]
file.close()

# Subsampling
sample_size = 2000
X, y = X[:sample_size], y[:sample_size]
```

In [2]:

```python
# Splitting the data into training and test sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [3]:

```python
# Importing svm model
from sklearn import svm

# Creating an svm Classifier
clf = svm.SVC()

# Train the model using the training sets
clf.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = clf.predict(X_test)
```

In [4]:

```python
# Import sckikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9275
```

`Question 7:` **Using the same (subsampled) training dataset used in the previous sections, employ `GridSearchCV` to find the best hyperparameter settings based on 5-fold cross-validation for a `RandomForestClassifier`. Consider `n_estimators` $\in \{50, 100, 200\}$ and `max_features` $\in \{0.1, 0.25\}$. Use the default values for the remaining hyperparameters. Compute the accuracy of the best model on the corresponding test dataset. Show the code in the report.**

In [7]:

```python
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

# Create a Gaussian Classifier
random_forest = RandomForestClassifier()

from sklearn.model_selection import GridSearchCV

n_estimators = [50, 100, 200]
max_features = [0.1, 0.25]

parameter_grid = dict(n_estimators = n_estimators,
              max_features=max_features)

random_forestCV = GridSearchCV(random_forest, parameter_grid, cv=5)
random_forestCV.fit(X_train, y_train)

print('Best hyperparameter setting: {0}.'.format(random_forestCV.best_estimator_))
print('Average accuracy across folds of best hyperparameter setting: {0}.'.format(random_forestCV.best_score_))
print('Test dataset accuracy of best hyperparameter setting: {0}.'.format(random_forestCV.score(X_test, y_test)))
```

```
Best hyperparameter setting: RandomForestClassifier(max_features=0.1, n_estimators=200).
Average accuracy across folds of best hyperparameter setting: 0.909375.
Test dataset accuracy of best hyperparameter setting: 0.92.
```