Titanic Passenger Survival Prediction

Student’s Name : ADIDEV-C

E-mail : adidev.c146@gmail.com

Submission Date : 28/09/2025

Objective

- **Extract** the informations from the dataset
- Find common **patterns** and **trends** shown by the data
- Find the **factors** affecting the survival of a passenger
- **Predict** the chance of survival of a new passenger using the given details
- **Cluster** the data and analyse the situations on all the clusters

Dataset Overview

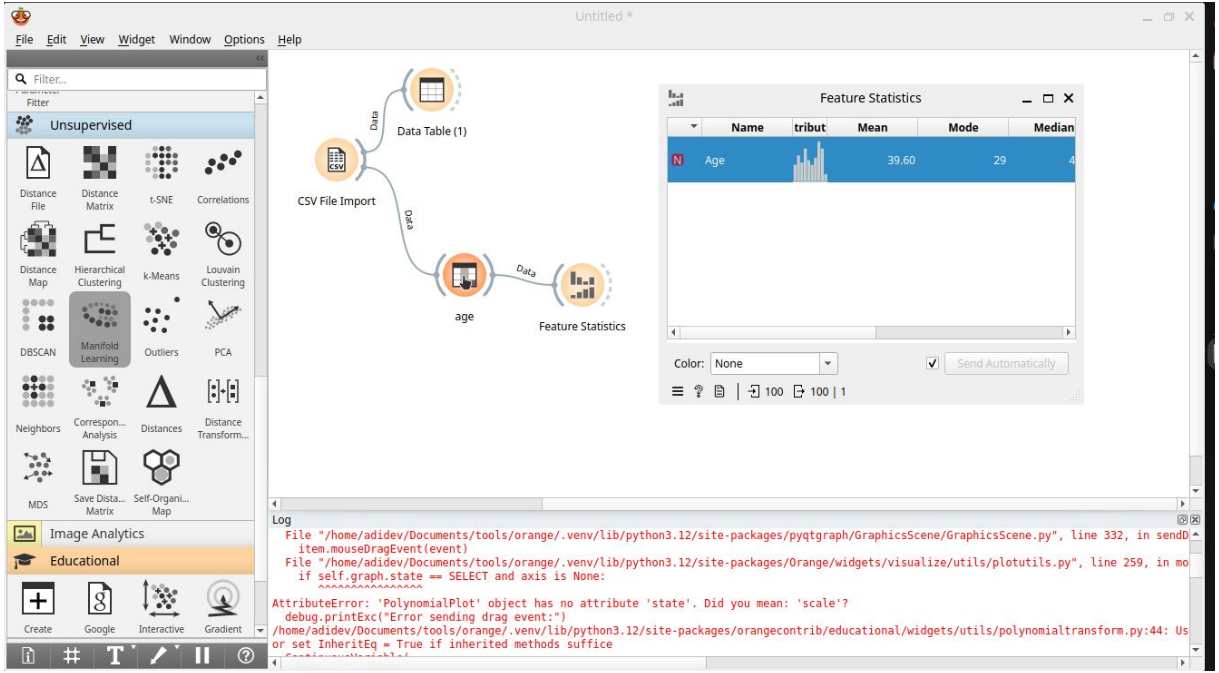
- The dataset contains the record of 100 passengers who either survived or not survived in the Titanic disaster
- Each of the passenger has 7 features, they are : PassengerID,Age,Gender,Pclass,Survived or not ,Embarked,TravelingAlone or not . etc
- The factors **NOT** affecting the state of survival of passenger are
  - PassengerID - just a unique id given to the passenger
- All the features except PassengerID plays a major role in the survival rate of the passenger
  - Age
  - Gender
  - PClass
  - Embarked
  - Was Travelling Alone

Tools & Techniques

- **Orange**
  - I used orange to do the Exploratory Data Analysis(EDA) on this project
  - Most of the EDA questions can be answered via plotting the data efficiently
- **Python**
  - I used python to apply the K-NN and K-Means algorithm to dataset
  - I felt python is more appropriate for applying K-NN and K-Means to the datasets which has more than 3 features
  - Libraries used:
    - Pandas
    - Numpy
- **Visual Studio Code**
  - To write the code for custom data analysis tools
- **Google Sheets & Libre Office Calc**
  - For viewing the data in table form, and exporting it to csv format

Exploratory Data Analysis (EDA)

1) Average age of passengers



Feature Statistics

	Name	tribut	Mean	Mode	Median
N	Age		39.60	29	4

Color: None

☑ Send Automatically

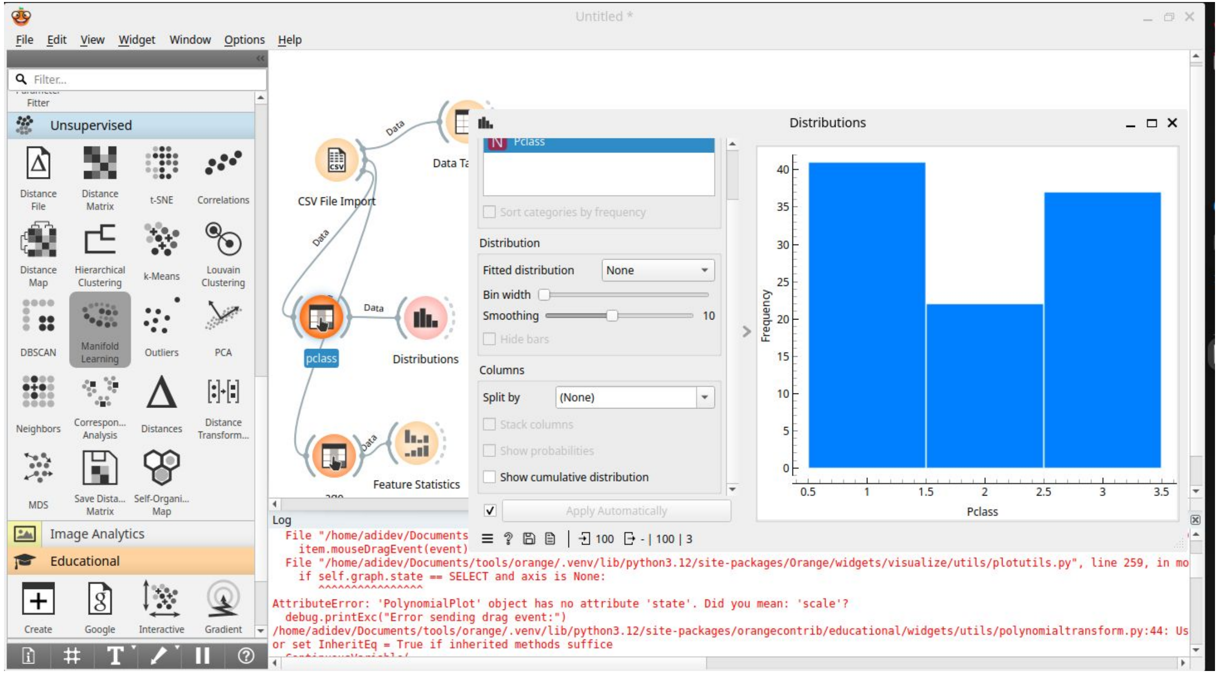
100 100 | 1

```
  File "/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/pyqtgraph/GraphicsScene/GraphicsScene.py", line 332, in sendD
    item.mouseDragEvent(event)
  File "/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/visualize/utils/plotutils.py", line 259, in mo
    if self.graph.state == SELECT and axis is None:
       ^^^^^^^^^^^^^^^^

AttributeError: 'PolynomialPlot' object has no attribute 'state'. Did you mean: 'scale'?
  debug.printExc("Error sending drag event:")
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/orangecontrib/educational/widgets/utils/polynomialtransform.py:44: Us
or set InheritEq = True if inherited methods suffice
```

When I selected the age column and applied the feature statistics, the mean was observed to be 39.6 we can round it to 40

MEAN AGE : 40

2) PClass with highest number of passengers



□ Sort categories by frequency

Distribution

Fitted distribution: None

Bin width

Smoothing: 10

□ Hide bars

Columns

Split by: (None)

□ Stack columns

□ Show probabilities

□ Show cumulative distribution

☑ Apply Automatically

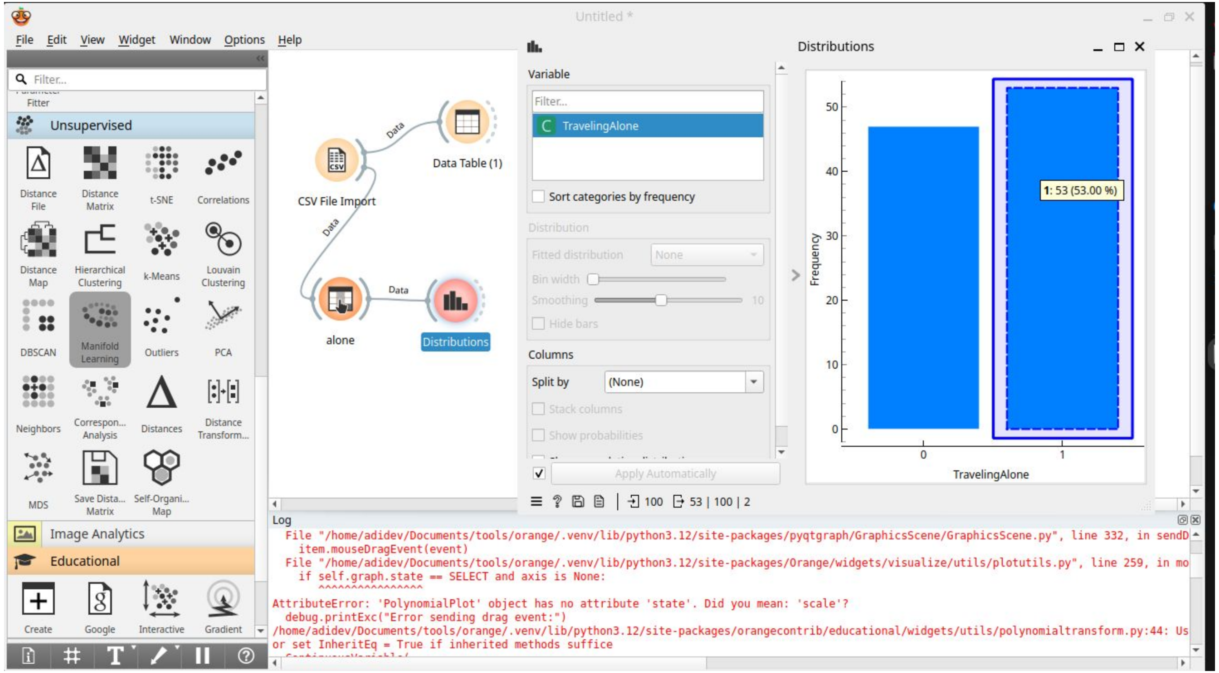
Log

```
  File "/home/adidev/Documents
    item.mouseDragEvent(event)
  File "/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/visualize/utils/plotutils.py", line 259, in mo
    if self.graph.state == SELECT and axis is None:
       ^^^^^^^^^^^^^^^^

AttributeError: 'PolynomialPlot' object has no attribute 'state'. Did you mean: 'scale'?
  debug.printExc("Error sending drag event:")
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/orangecontrib/educational/widgets/utils/polynomialtransform.py:44: Us
or set InheritEq = True if inherited methods suffice
```

When I selected the class column via select columns widget and observed its distributions, I understood that class 1 has the highest number of passengers

CLASS WITH HIGHEST NUMBER OF PASSENGERS : CLASS 1

3) Number of passengers travelling alone

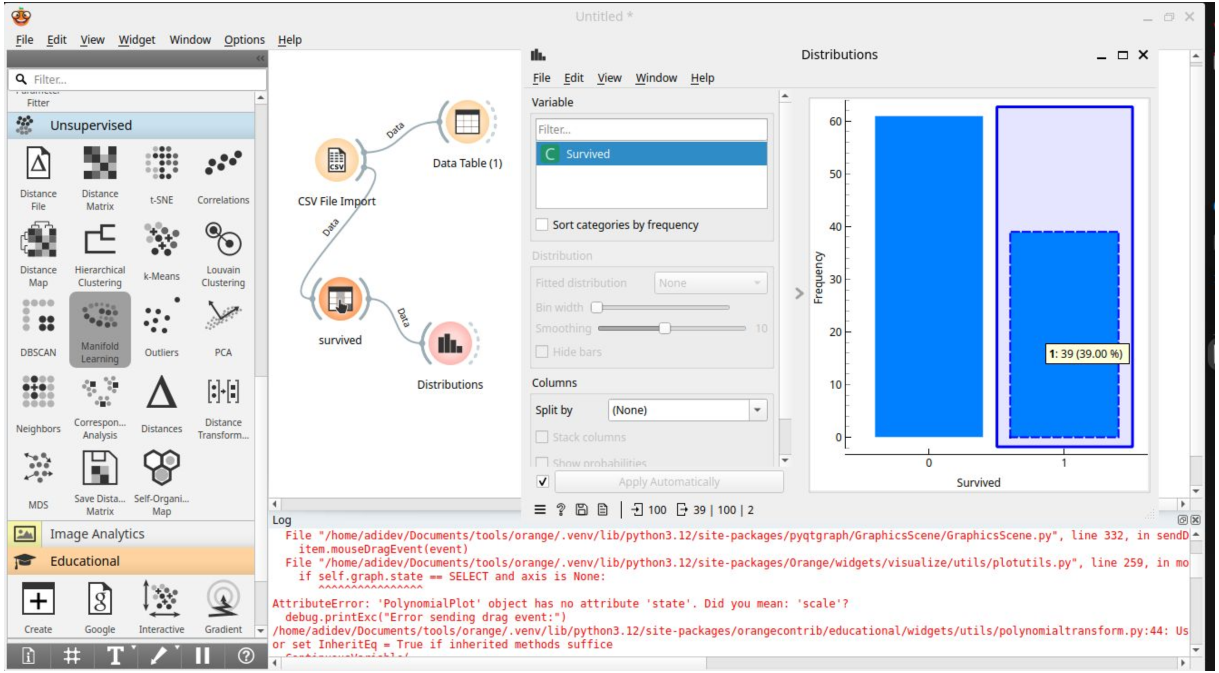


```
  File "/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/pyqtgraph/GraphicsScene/GraphicsScene.py", line 332, in sendD
    item.mouseDragEvent(event)
  File "/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/visualize/utils/plotutils.py", line 259, in mo
    if self.graph.state == SELECT and axis is None:
       ^^^^^^^^^^^^^^^^
AttributeError: 'PolynomialPlot' object has no attribute 'state'. Did you mean: 'scale'?
  debug.printExc("Error sending drag event:")
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/orangecontrib/educational/widgets/utils/polynomialtransform.py:44: Us
or set InheritEq = True if inherited methods suffice
```

I selected the ‘traveling alone’ column with the select column widget and observed their distributions, its observed that 53 passengers were traveling alone

NUMBER OF PASSENGERS TRAVELLING ALONE : 53

4) Percentage of passengers survived



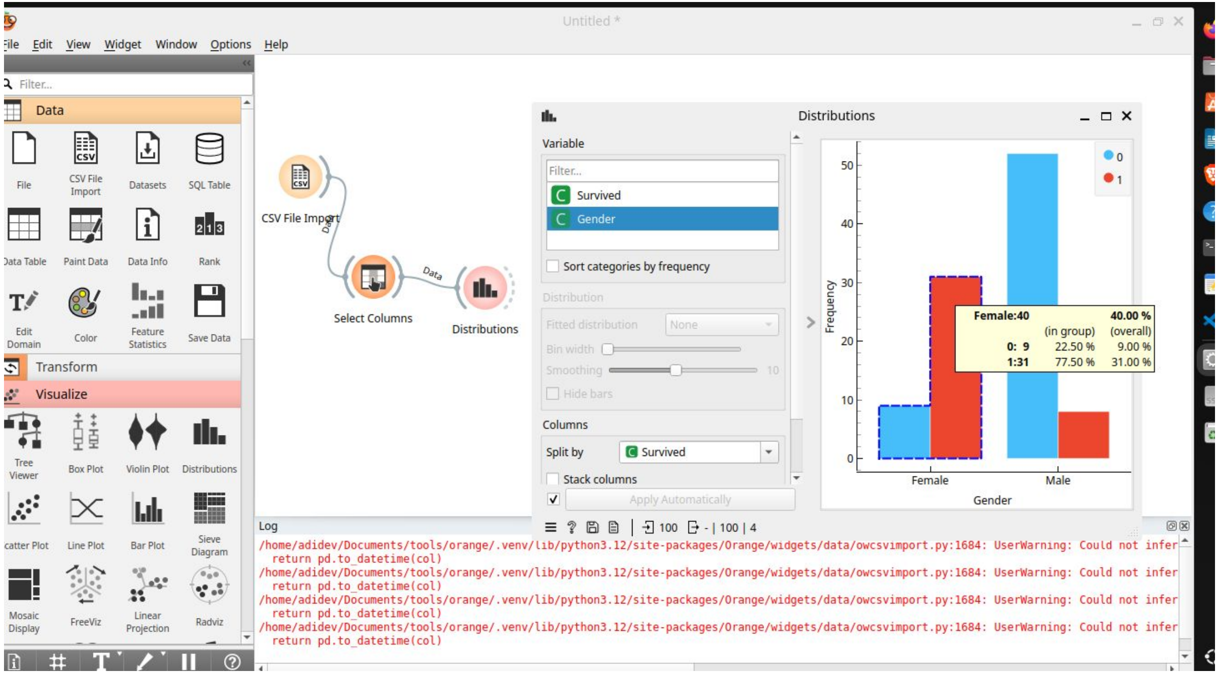
```
  File "/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/pyqtgraph/GraphicsScene/GraphicsScene.py", line 332, in sendD
    item.mouseDragEvent(event)
  File "/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/visualize/utils/plotutils.py", line 259, in mo
    if self.graph.state == SELECT and axis is None:
       ^^^^^^^^^^^^^^^^

AttributeError: 'PolynomialPlot' object has no attribute 'state'. Did you mean: 'scale'?
  debug.printExc("Error sending drag event:")
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/orangecontrib/educational/widgets/utils/polynomialtransform.py:44: Us
or set InheritEq = True if inherited methods suffice
```

I selected the survival chance column and observed its distributions.

It is observed that 39% of the total passengers survived the disaster

PERCENTAGE OF PASSENGERS SURVIVED : 39%

5) Group with better chances of survival(male/female)



Variable

Filter...

- Survived
- Gender

☐ Sort categories by frequency

Distribution

Fitted distribution: None

Bin width

Smoothing 10

☐ Hide bars

Columns

Split by: Survived

☐ Stack columns

☑ Apply Automatically

100 | - | 100 | 4

```
Female:40                 40.00 %
              (in group)  (overall)
   0: 9        22.50 %     9.00 %
   1:31        77.50 %    31.00 %
```

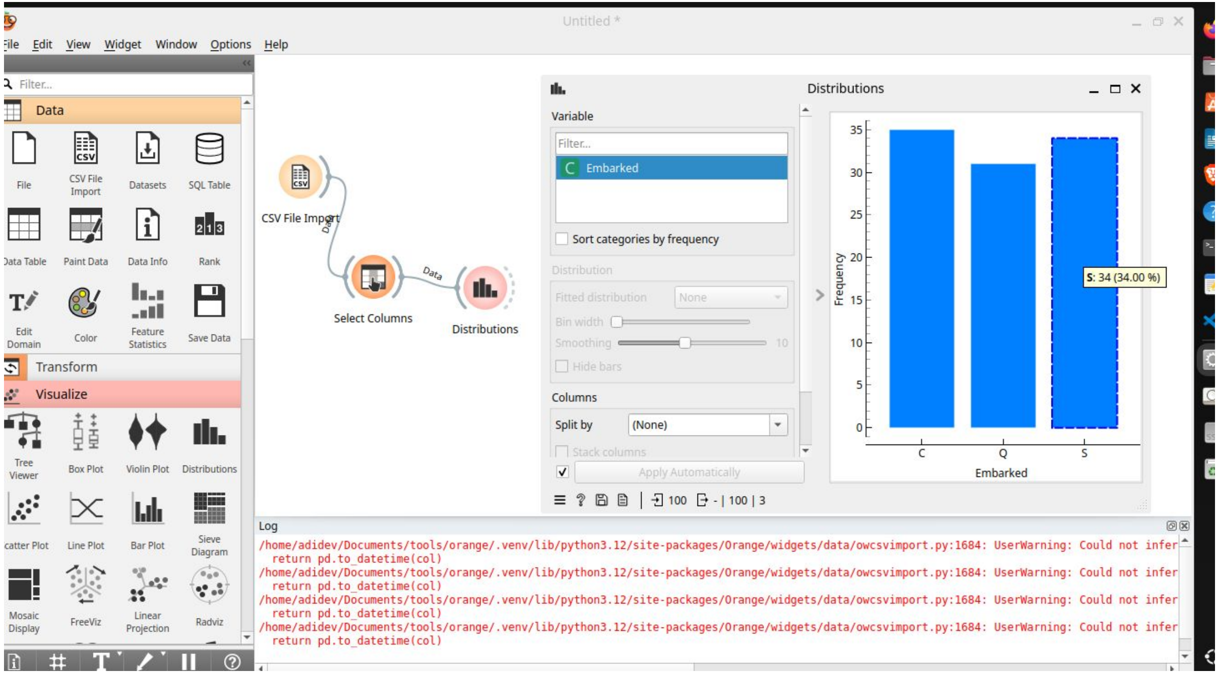
```
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/data/owcsvimport.py:1684: UserWarning: Could not infer
  return pd.to_datetime(col)
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/data/owcsvimport.py:1684: UserWarning: Could not infer
  return pd.to_datetime(col)
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/data/owcsvimport.py:1684: UserWarning: Could not infer
  return pd.to_datetime(col)
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/data/owcsvimport.py:1684: UserWarning: Could not infer
  return pd.to_datetime(col)
```

I observed the distributions of the **survived** and **gender** columns

In that distributions, I split the graph by the **‘survived’** feature. It is observed that, females had the better chance of survival

GROUP WITH BETTER CHANCE OF SURVIVAL : FEMALES

6) Number of passengers embarked from each place



Variable: Embarked

☐ Sort categories by frequency

Split by: (None)

☑ Apply Automatically

```
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/data/owcsvimport.py:1684: UserWarning: Could not infer
  return pd.to_datetime(col)
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/data/owcsvimport.py:1684: UserWarning: Could not infer
  return pd.to_datetime(col)
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/data/owcsvimport.py:1684: UserWarning: Could not infer
  return pd.to_datetime(col)
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/data/owcsvimport.py:1684: UserWarning: Could not infer
  return pd.to_datetime(col)
```

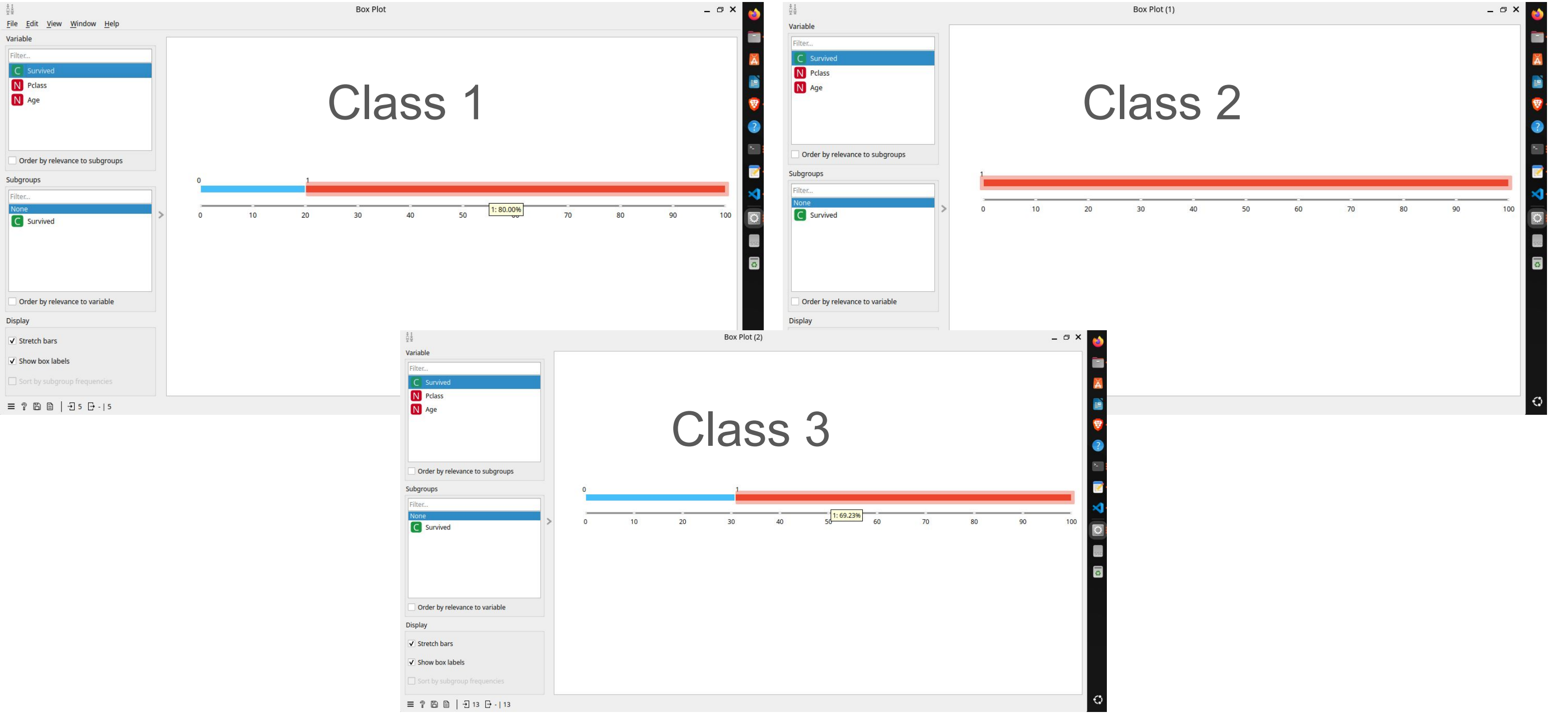
I observed the distributions of **Embarked** feature.

Number of passengers embarked from Southampton : 34

Number of passengers embarked from Queenstown : 31

Number of passengers embarked from Cherbourg : 35

7) Which passenger class had the highest survival rate - below



Box Plot

File Edit View Window Help

Variable

Filter...

C Survived

N Pclass

N Age

□ Order by relevance to subgroups

Subgroups

Filter...

None

C Survived

□ Order by relevance to variable

Display

☑ Stretch bars

☑ Show box labels

□ Sort by subgroup frequencies

Class 1

1: 80.00%

Box Plot (1)

Variable

Filter...

C Survived

N Pclass

N Age

□ Order by relevance to subgroups

Subgroups

Filter...

None

C Survived

□ Order by relevance to variable

Display

Class 2

Box Plot (2)

Variable

Filter...

C Survived

N Pclass

N Age

□ Order by relevance to subgroups

Subgroups

Filter...

None

C Survived

□ Order by relevance to variable

Display

☑ Stretch bars

☑ Show box labels

□ Sort by subgroup frequencies

Class 3

1: 69.23%

When i checked the survival rate of each class using box plot,its clear that 2nd class had the highest survival rate for passengers below 18

PASSENGER CLASS WITH HIGHEST SURVIVAL RATE(BELOW 18) : CLASS 2

8) Which combination had the best chance of survival

To find this, I needed to go through

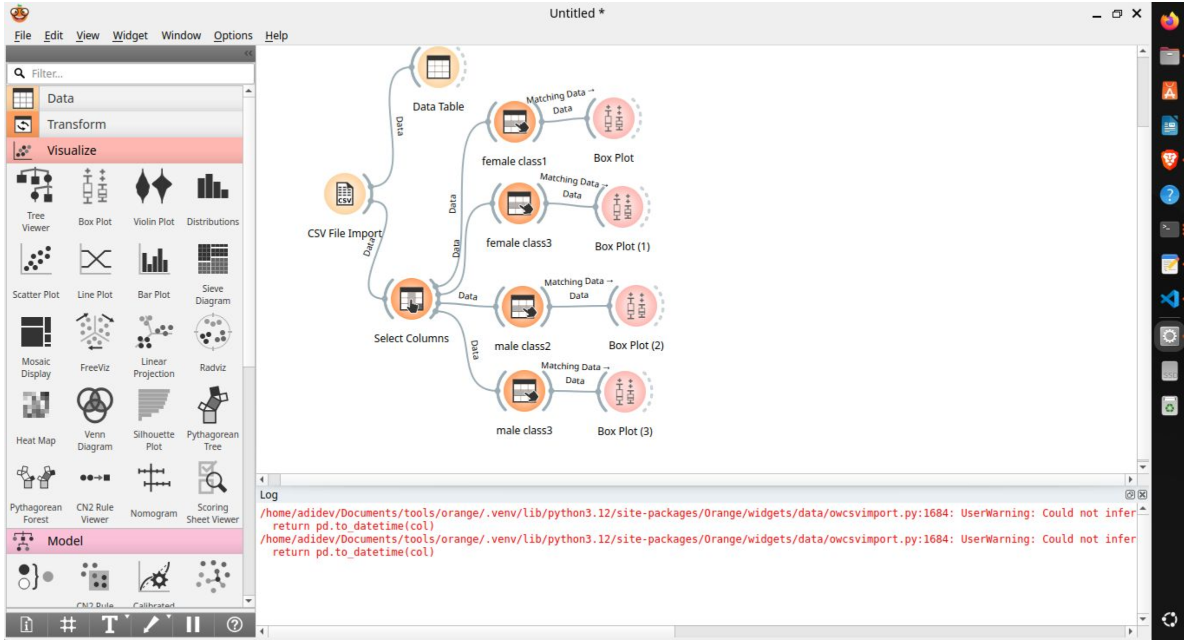
all 4 groups

1)Female, Class 1

2)Female, Class 3

3)Male, Class 2

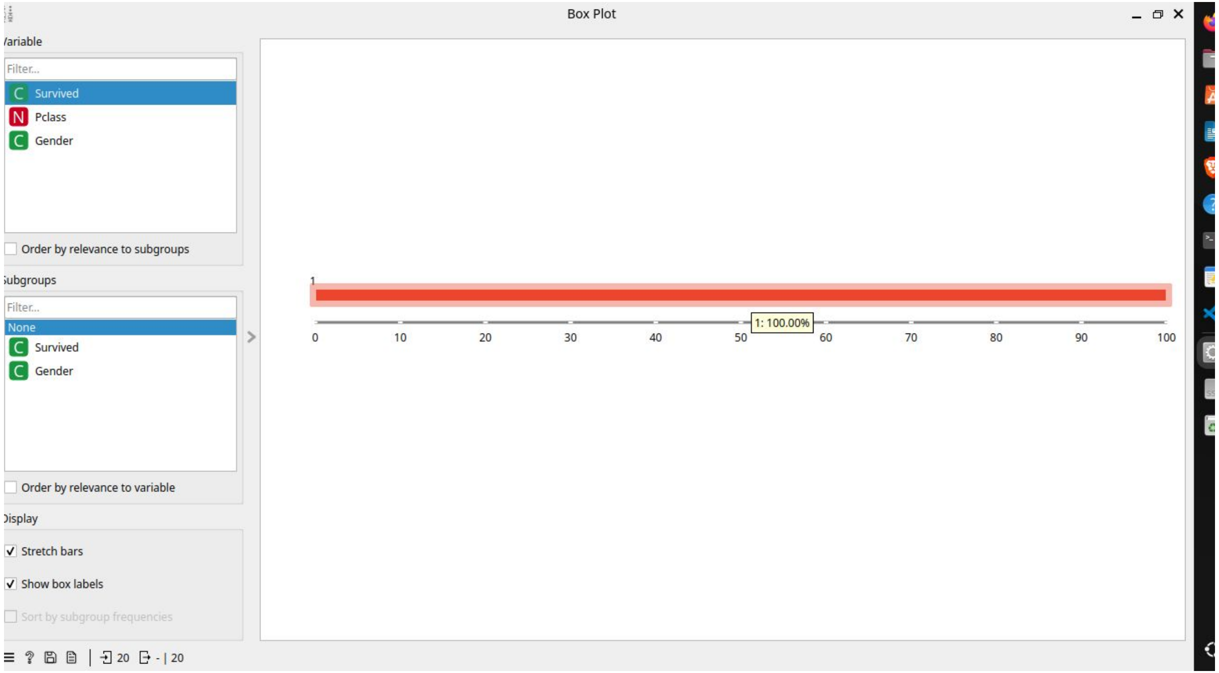
4)Male, Class 3



```
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/data/owcsvimport.py:1684: UserWarning: Could not infer
  return pd.to_datetime(col)
/home/adidev/Documents/tools/orange/.venv/lib/python3.12/site-packages/Orange/widgets/data/owcsvimport.py:1684: UserWarning: Could not infer
  return pd.to_datetime(col)
```

So i created 4 branches from the root dataset with the conditions to select the rows and visualised their survival rate using box-plot

Female, Class 1


Box Plot

ariable

Filter...

- Survived
- Pclass
- Gender

□ Order by relevance to subgroups

ubgroups

Filter...

- None
- Survived
- Gender

□ Order by relevance to variable

isplay

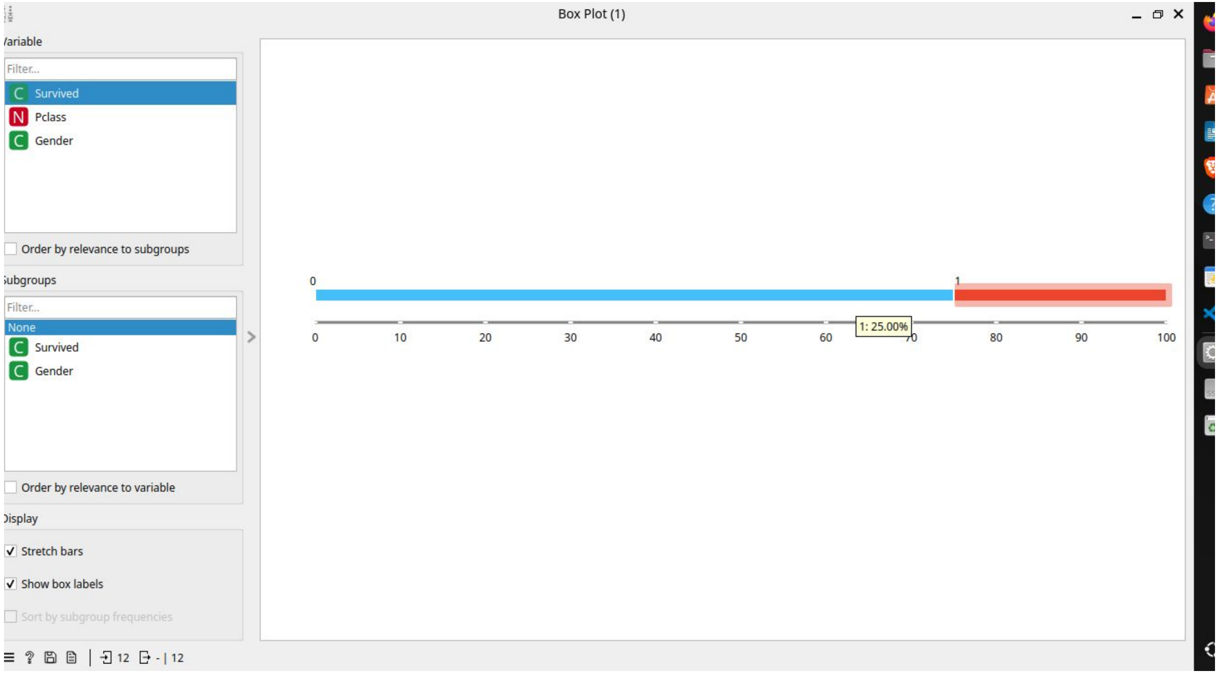
☑ Stretch bars

☑ Show box labels

□ Sort by subgroup frequencies

CHANCE OF SURVIVAL: 100%

Female, Class 3



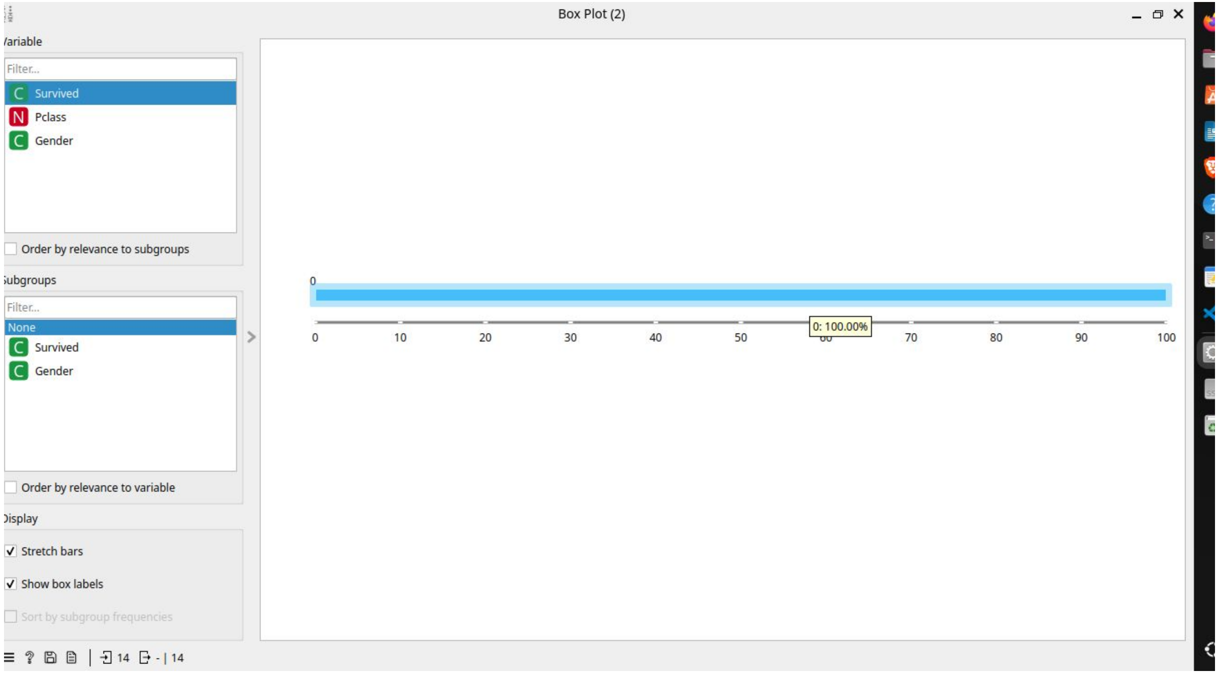
☑ Stretch bars

☑ Show box labels

□ Sort by subgroup frequencies

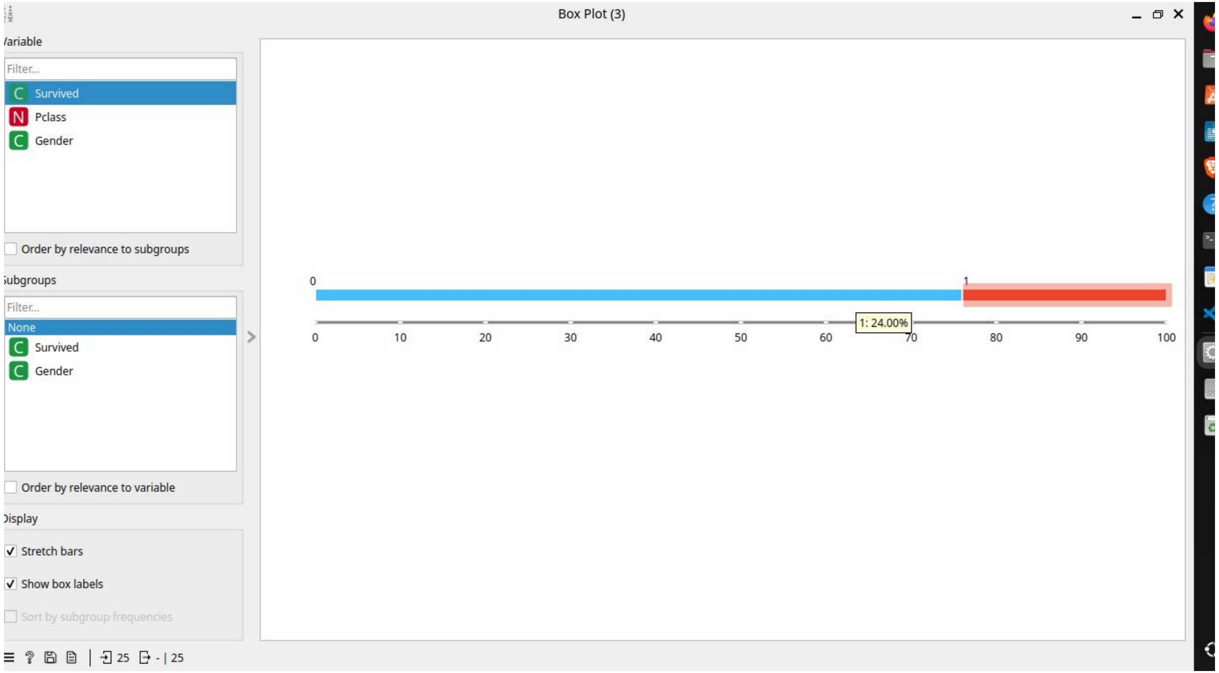
CHANCE OF SURVIVAL: 25%

Male, Class 2



CHANCE OF SURVIVAL: 0%

Male, Class 3


Box Plot (3)

/ariable

Filter...

C Survived

N Pclass

C Gender

□ Order by relevance to subgroups

;ubgroups

Filter...

None

C Survived

C Gender

□ Order by relevance to variable

)isplay

☑ Stretch bars

☑ Show box labels

□ Sort by subgroup frequencies

25 - | 25

CHANCE OF SURVIVAL:24%

Result

From the above results,

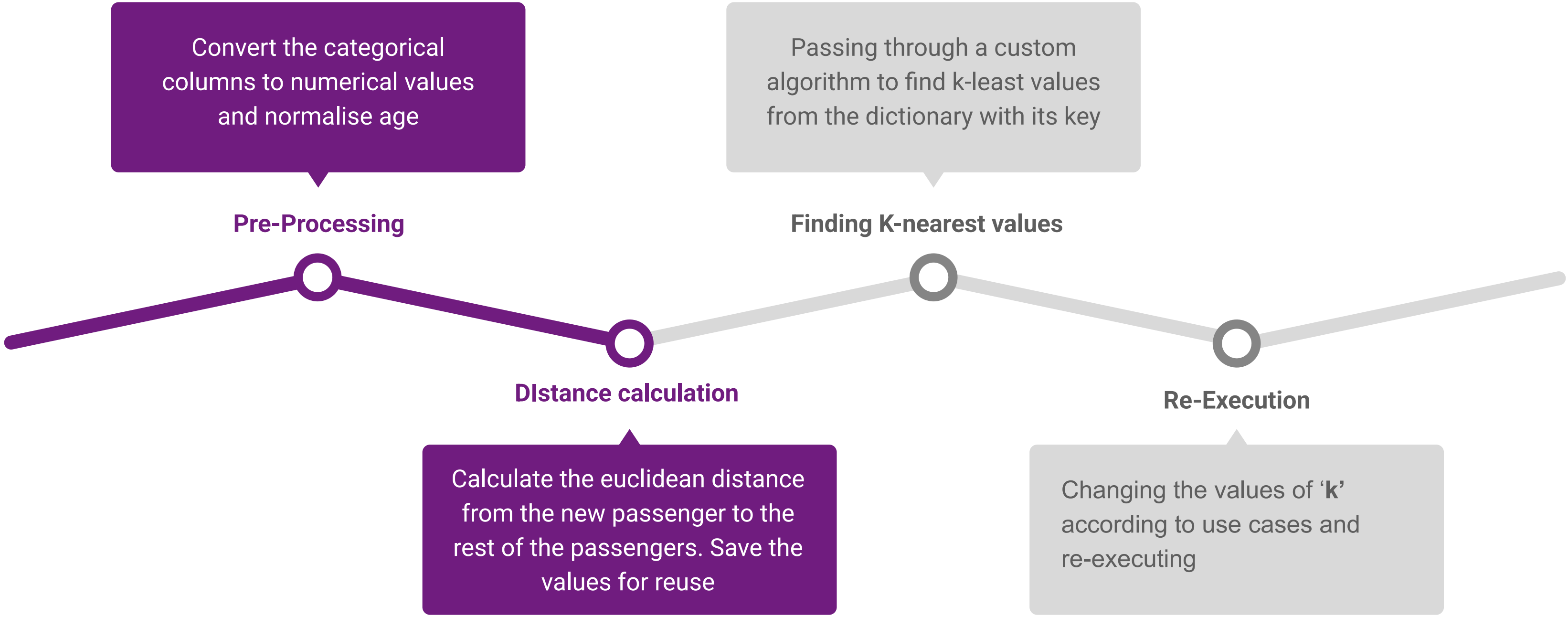
We can understand that the the combination **“female-class1”** had the best chance of survival

Methodology

K-NN Classification

- I used python for applying K-NN to the dataset
- I created a python script that compares the 101th passenger to the rest of them and calculates the Euclidean distance and save them to a JSON file
- I converted the categorical columns into numerical values using a map feature on pandas and also normalised the ages
- Then I used the JSON file as an input for another python function which returns the least **‘k’** values from a python dictionary including its key
- Using the key, I just grabbed those index from the csv file using pandas and displayed it on the screen.
- Using the above python function, I was able to find the answers to the data by changing the values of **‘k’**

K-NN Classification - Custom algorithm



Pre-Processing

Convert the categorical columns to numerical values and normalise age

DIstance calculation

Calculate the euclidean distance from the new passenger to the rest of the passengers. Save the values for reuse

Finding K-nearest values

Passing through a custom algorithm to find k-least values from the dictionary with its key

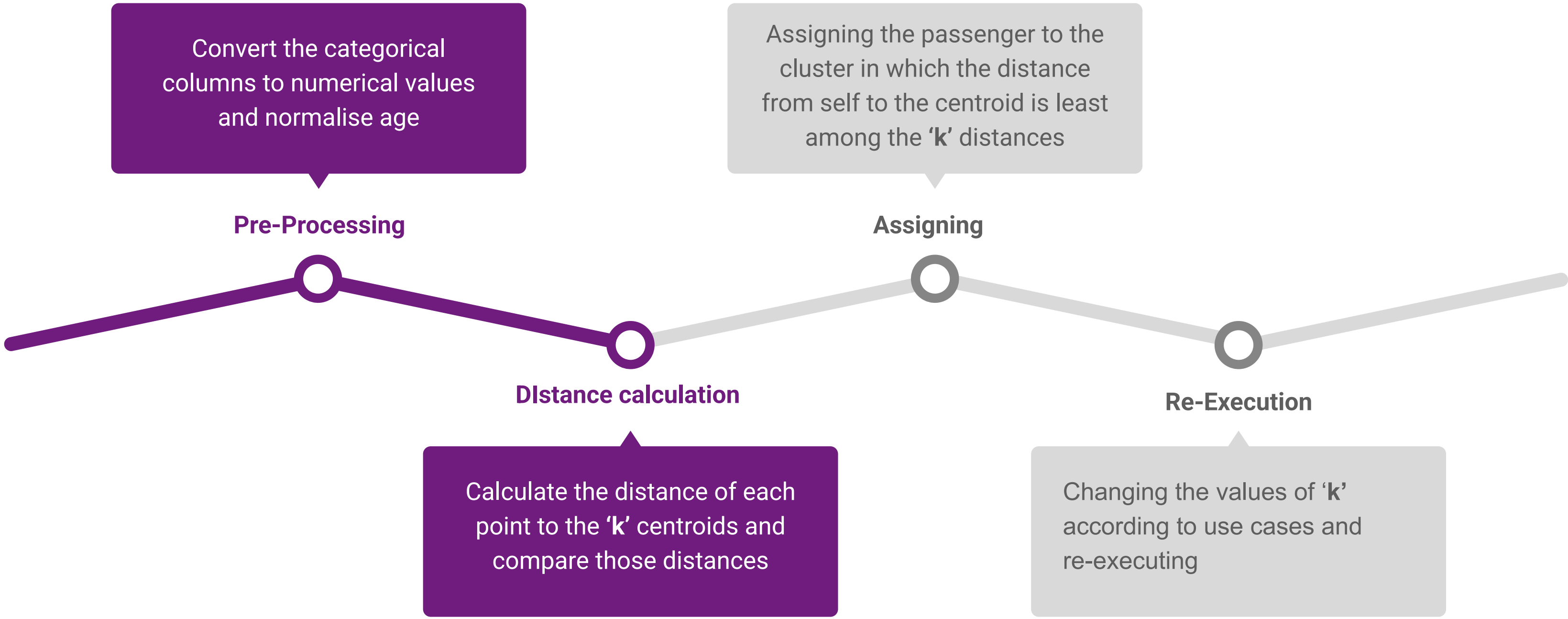
Re-Execution

Changing the values of ‘**k**’ according to use cases and re-executing

K-Means Clustering

- I used python for applying K-Means to the dataset
- I searched the github,stackoverflow etc to find existing K-Means algorithms which allow us to fix the centroids for the clusters. But i found nothing. So I made my own algorithm to cluster the data
- First i did the preprocessing steps such as normalising age and converting categorical columns to numerical values
- Then i fixed the centroids and classified all points to cluster1 and cluster2 and stored them in a dictionary in the form {“passenger id”:cluster}
- Then I used that dictionary to understand the dataset
- I also exported those dictionaries to JSON format, to be useful in case of manual reuse

K-Means Clustering - Custom algorithm



Pre-Processing

Convert the categorical columns to numerical values and normalise age

DIstance calculation

Calculate the distance of each point to the ‘**k**’ centroids and compare those distances

Assigning

Assigning the passenger to the cluster in which the distance from self to the centroid is least among the ‘**k**’ distances

Re-Execution

Changing the values of ‘**k**’ according to use cases and re-executing

K-NN Classification Results

3 nearest neighbors

After running the python script for K-NN with K=3
The nearest points around the point 101 are observed to be

- **26 : distance = 1.41 unit**
- **71 : distance = 1.41 unit**
- **80 : distance = 1.41 unit**

All three points were in the same distance from the reference point

Survival prediction K=5

After running the python script for K-NN with K=5
The nearest points around the point 101 are observed to be

- **26 : distance = 1.41 unit; survived = 1**
- **71 : distance = 1.41 unit; survived = 0**
- **80 : distance = 1.41 unit; survived = 0**
- **41 : distance = 1.73 unit; survived = 1**
- **32 : distance = 2.0 unit; survived = 0**

No. of survived = 2
No. of not survived = 3

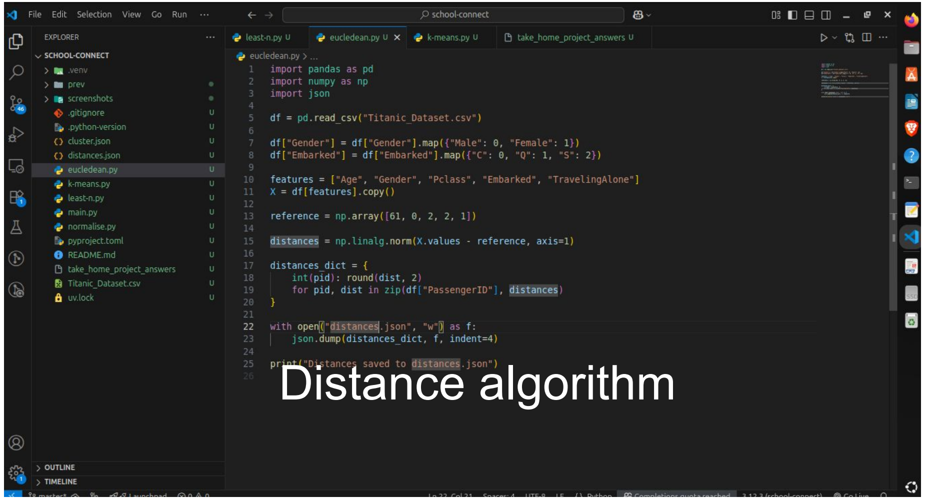
ACCORDING TO K-NN ALGORITHM, THE 101TH PASSENGER WILL NOT SURVIVE
SURVIVAL STATUS = 0

Nearest survivals k=9

After running the python script for K-NN with K=9
The nearest points around the point 101 are observed to be

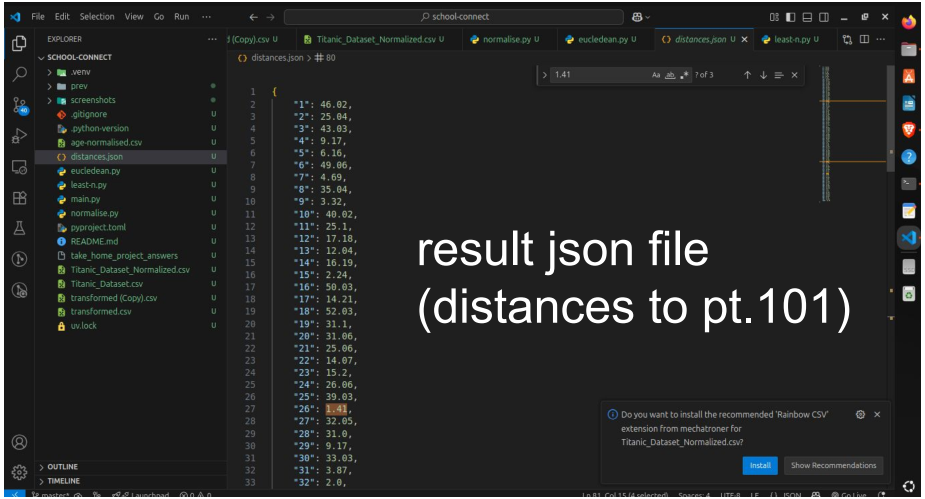
- **26 : distance = 1.41 unit; survived = 1**
- **71 : distance = 1.41 unit; survived = 0**
- **80 : distance = 1.41 unit; survived = 0**
- **41 : distance = 1.73 unit; survived = 1**
- **32 : distance = 2.0 unit; survived = 0**
- **57 : distance = 2.0 unit; survived = 1**
- **92 : distance = 2.0 unit; survived = 0**
- **15 : distance = 2.24 unit; survived = 1**
- **69 : distance = 2.45 unit; survived = 1**

No. of survived = 5
No. of not survived = 4



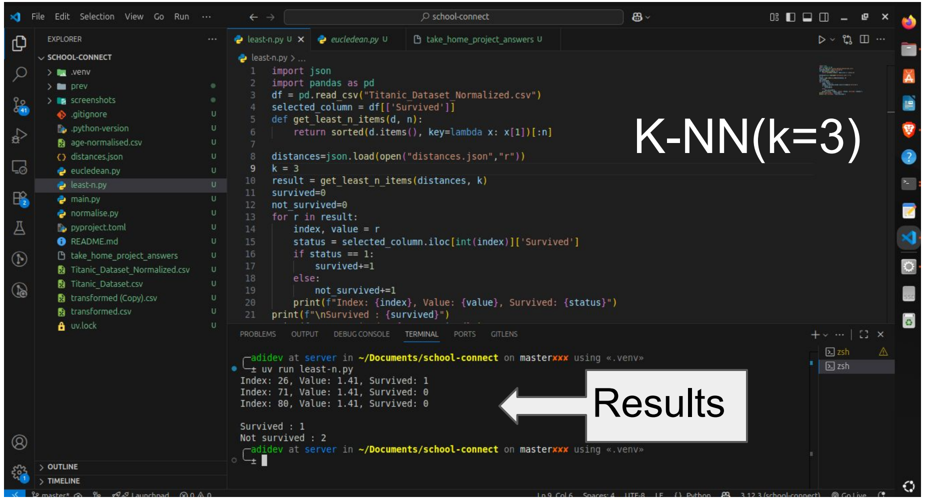
Distance algorithm

```python
import pandas as pd
import numpy as np
import json

df = pd.read_csv("Titanic_Dataset.csv")

df["Gender"] = df["Gender"].map({"Male": 0, "Female": 1})
df["Embarked"] = df["Embarked"].map({"C": 0, "Q": 1, "S": 2})

features = ["Age", "Gender", "Pclass", "Embarked", "TravelingAlone"]
X = df[features].copy()

reference = np.array([61, 0, 2, 2, 1])

distances = np.linalg.norm(X.values - reference, axis=1)

distances_dict = {
    int(pid): round(dist, 2)
    for pid, dist in zip(df["PassengerID"], distances)
}

with open("distances.json", "w") as f:
    json.dump(distances_dict, f, indent=4)

print("Distances saved to distances.json")
```



result json file (distances to pt.101)

```json
{
    "1": 46.02,
    "2": 25.04,
    "3": 43.03,
    "4": 9.17,
    "5": 6.16,
    "6": 49.06,
    "7": 4.69,
    "8": 35.04,
    "9": 3.32,
    "10": 40.02,
    "11": 25.1,
    "12": 17.18,
    "13": 12.04,
    "14": 16.19,
    "15": 2.24,
    "16": 50.03,
    "17": 14.21,
    "18": 52.03,
    "19": 31.1,
    "20": 31.06,
    "21": 25.06,
    "22": 14.07,
    "23": 15.2,
    "24": 26.06,
    "25": 39.03,
    "26": 1.41,
    "27": 32.05,
    "28": 31.0,
    "29": 9.17,
    "30": 33.03,
    "31": 3.87,
    "32": 2.0,
```

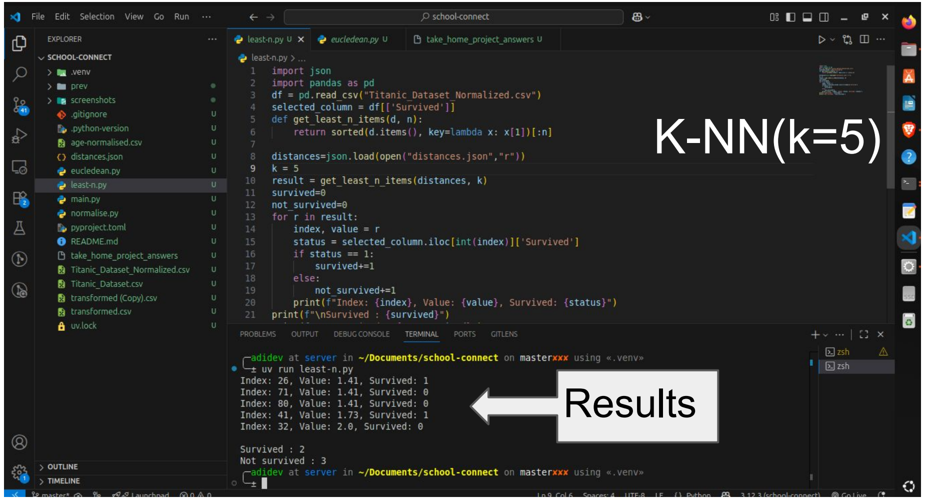


K-NN(k=3)

```python
import json
import pandas as pd
df = pd.read_csv("Titanic_Dataset_Normalized.csv")
selected_column = df[['Survived']]
def get_least_n_items(d, n):
    return sorted(d.items(), key=lambda x: x[1])[:n]

distances=json.load(open("distances.json","r"))
k = 3
result = get_least_n_items(distances, k)
survived=0
not_survived=0
for r in result:
    index, value = r
    status = selected_column.iloc[int(index)]['Survived']
    if status == 1:
        survived+=1
    else:
        not_survived+=1
    print(f"Index: {index}, Value: {value}, Survived: {status}")
print(f"\nSurvived : {survived}")
```

```
╭─adidev at server in ~/Documents/school-connect on masterxxx using «.venv»
╰─± uv run least-n.py
Index: 26, Value: 1.41, Survived: 1
Index: 71, Value: 1.41, Survived: 0
Index: 80, Value: 1.41, Survived: 0

Survived : 1
Not survived : 2
╭─adidev at server in ~/Documents/school-connect on masterxxx using «.venv»
╰─±
```

Results

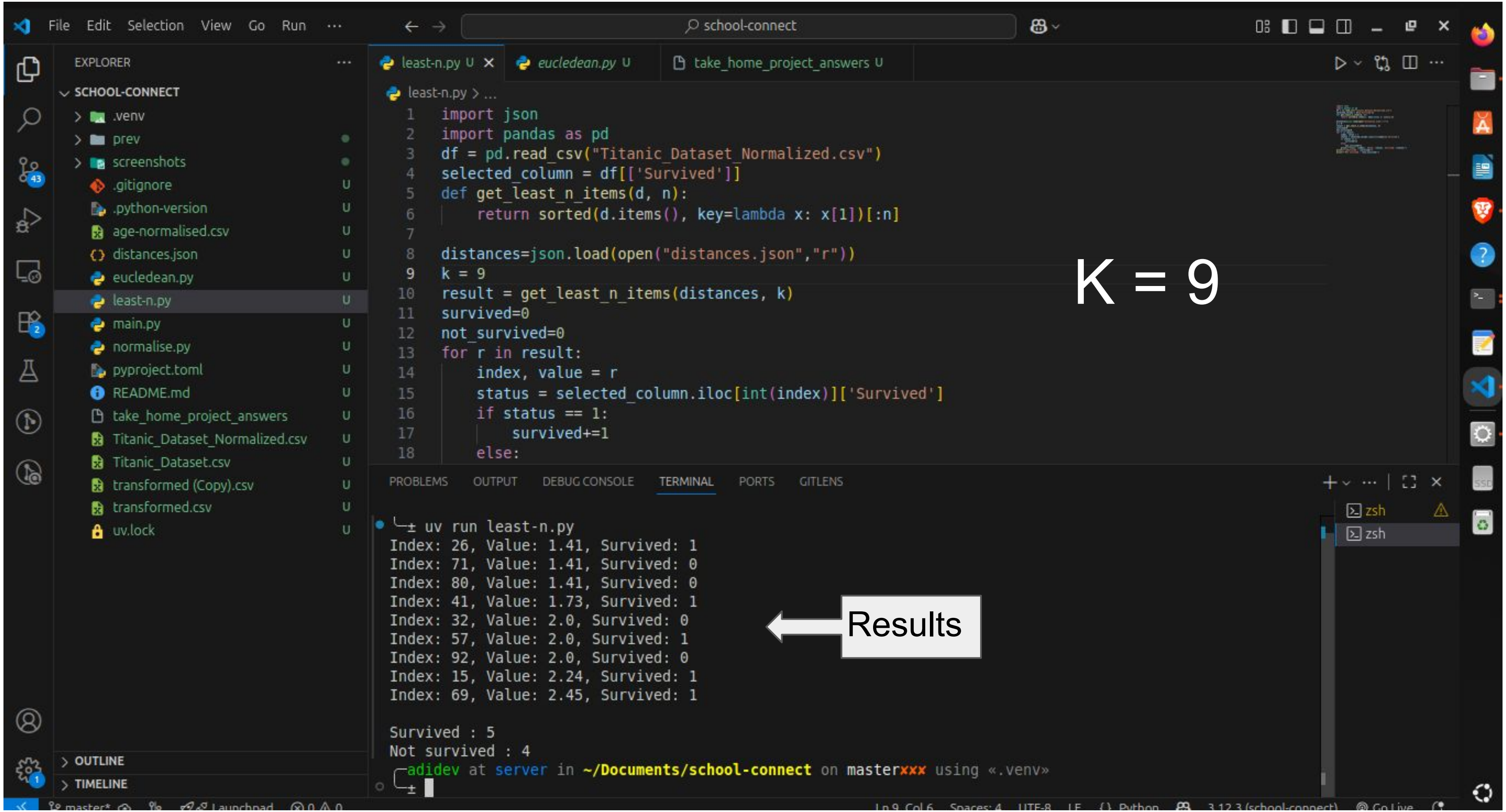


K-NN(k=5)

```python
import json
import pandas as pd
df = pd.read_csv("Titanic_Dataset_Normalized.csv")
selected_column = df[['Survived']]
def get_least_n_items(d, n):
    return sorted(d.items(), key=lambda x: x[1])[:n]

distances=json.load(open("distances.json","r"))
k = 5
result = get_least_n_items(distances, k)
survived=0
not_survived=0
for r in result:
    index, value = r
    status = selected_column.iloc[int(index)]['Survived']
    if status == 1:
        survived+=1
    else:
        not_survived+=1
    print(f"Index: {index}, Value: {value}, Survived: {status}")
print(f"\nSurvived : {survived}")
```

```
╭─adidev at server in ~/Documents/school-connect on masterxxx using «.venv»
╰─± uv run least-n.py
Index: 26, Value: 1.41, Survived: 1
Index: 71, Value: 1.41, Survived: 0
Index: 80, Value: 1.41, Survived: 0
Index: 41, Value: 1.73, Survived: 1
Index: 32, Value: 2.0, Survived: 0

Survived : 2
Not survived : 3
╭─adidev at server in ~/Documents/school-connect on masterxxx using «.venv»
╰─±
```

Results



```python
import json
import pandas as pd
df = pd.read_csv("Titanic_Dataset_Normalized.csv")
selected_column = df[['Survived']]
def get_least_n_items(d, n):
    return sorted(d.items(), key=lambda x: x[1])[:n]

distances=json.load(open("distances.json","r"))
k = 9
result = get_least_n_items(distances, k)
survived=0
not_survived=0
for r in result:
    index, value = r
    status = selected_column.iloc[int(index)]['Survived']
    if status == 1:
        survived+=1
    else:
```

K = 9

```
└± uv run least-n.py
Index: 26, Value: 1.41, Survived: 1
Index: 71, Value: 1.41, Survived: 0
Index: 80, Value: 1.41, Survived: 0
Index: 41, Value: 1.73, Survived: 1
Index: 32, Value: 2.0, Survived: 0
Index: 57, Value: 2.0, Survived: 1
Index: 92, Value: 2.0, Survived: 0
Index: 15, Value: 2.24, Survived: 1
Index: 69, Value: 2.45, Survived: 1

Survived : 5
Not survived : 4
┌adidev at server in ~/Documents/school-connect on masterxxx using «.venv»
└±
```

Results

K-Means Clustering Results

Passenger 99, Cluster

After running the python script for K-Means with K=2
The passenger 99 observed to be in cluster 2 (centroid = passenger 46)

Distance to passenger 4 = **2.83 (Cluster 1)**
Distance to passenger 46 = **2.03 units (Cluster 2)**

Passenger 99 is in Cluster 2 (centroid = passenger 46)

Distance : 9 <-> 46

Passenger 9 (pre-processed values) : 1,1,1,2,1,0.8

Passenger 46 (pre-processed values) : 1,3,1,2,1,0.0

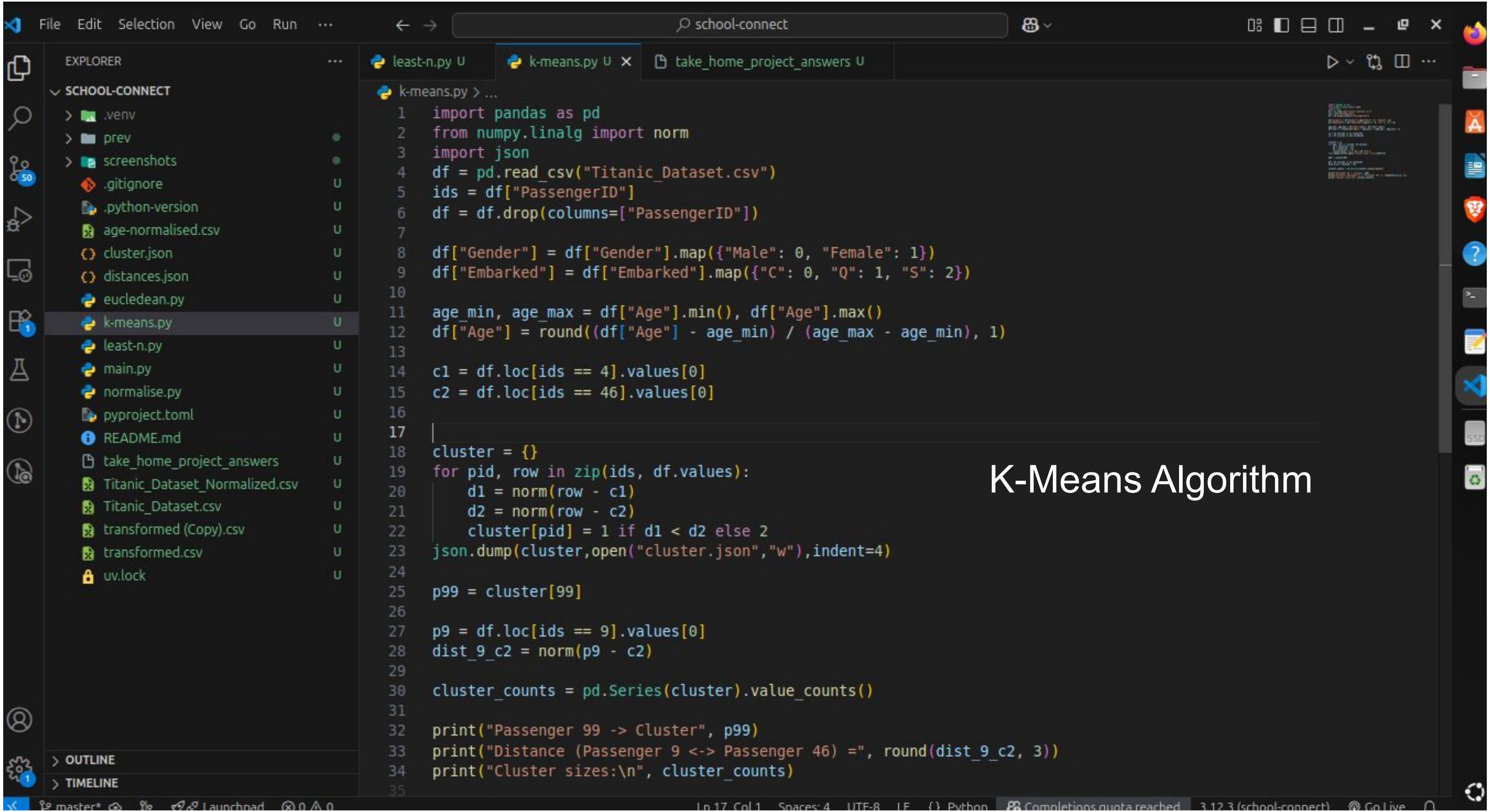
$$\text{Distance} = \sqrt{((1-1)^2+(1-3)^2+(1-1)^2+(2-2)^2+(1-1)^2+(0.8-0)^2)}$$

Distance between passenger 9 and 46 = 2.15 units

Cluster with more passengers

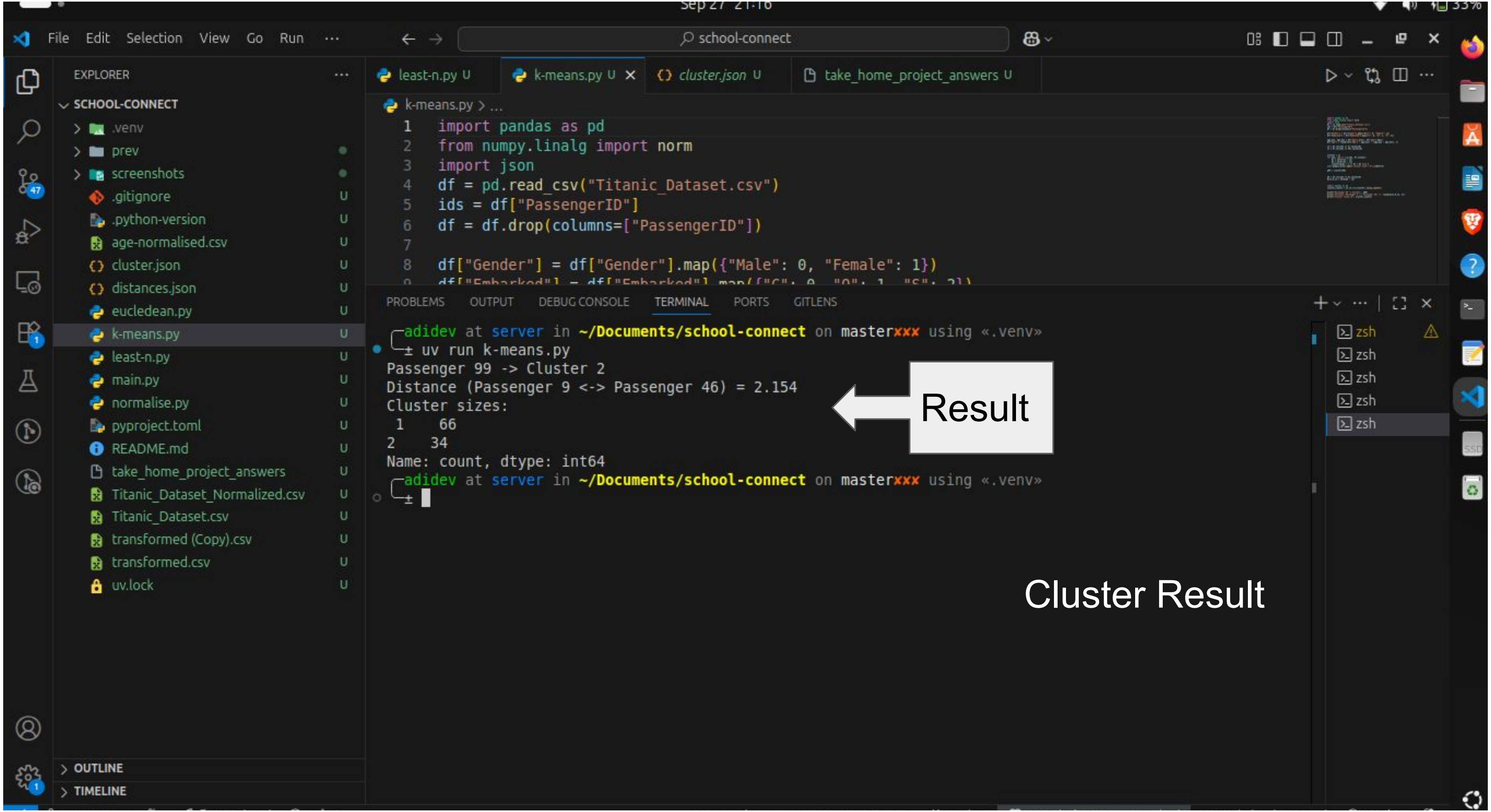
Cluster 1 : 66 passengers
Cluster 2 : 34 passengers

Cluster 1 has more passengers



```python
import pandas as pd
from numpy.linalg import norm
import json
df = pd.read_csv("Titanic_Dataset.csv")
ids = df["PassengerID"]
df = df.drop(columns=["PassengerID"])

df["Gender"] = df["Gender"].map({"Male": 0, "Female": 1})
df["Embarked"] = df["Embarked"].map({"C": 0, "Q": 1, "S": 2})

age_min, age_max = df["Age"].min(), df["Age"].max()
df["Age"] = round((df["Age"] - age_min) / (age_max - age_min), 1)

c1 = df.loc[ids == 4].values[0]
c2 = df.loc[ids == 46].values[0]


cluster = {}
for pid, row in zip(ids, df.values):
    d1 = norm(row - c1)
    d2 = norm(row - c2)
    cluster[pid] = 1 if d1 < d2 else 2
json.dump(cluster,open("cluster.json","w"),indent=4)

p99 = cluster[99]

p9 = df.loc[ids == 9].values[0]
dist_9_c2 = norm(p9 - c2)

cluster_counts = pd.Series(cluster).value_counts()

print("Passenger 99 -> Cluster", p99)
print("Distance (Passenger 9 <-> Passenger 46) =", round(dist_9_c2, 3))
print("Cluster sizes:\n", cluster_counts)
```

K-Means Algorithm

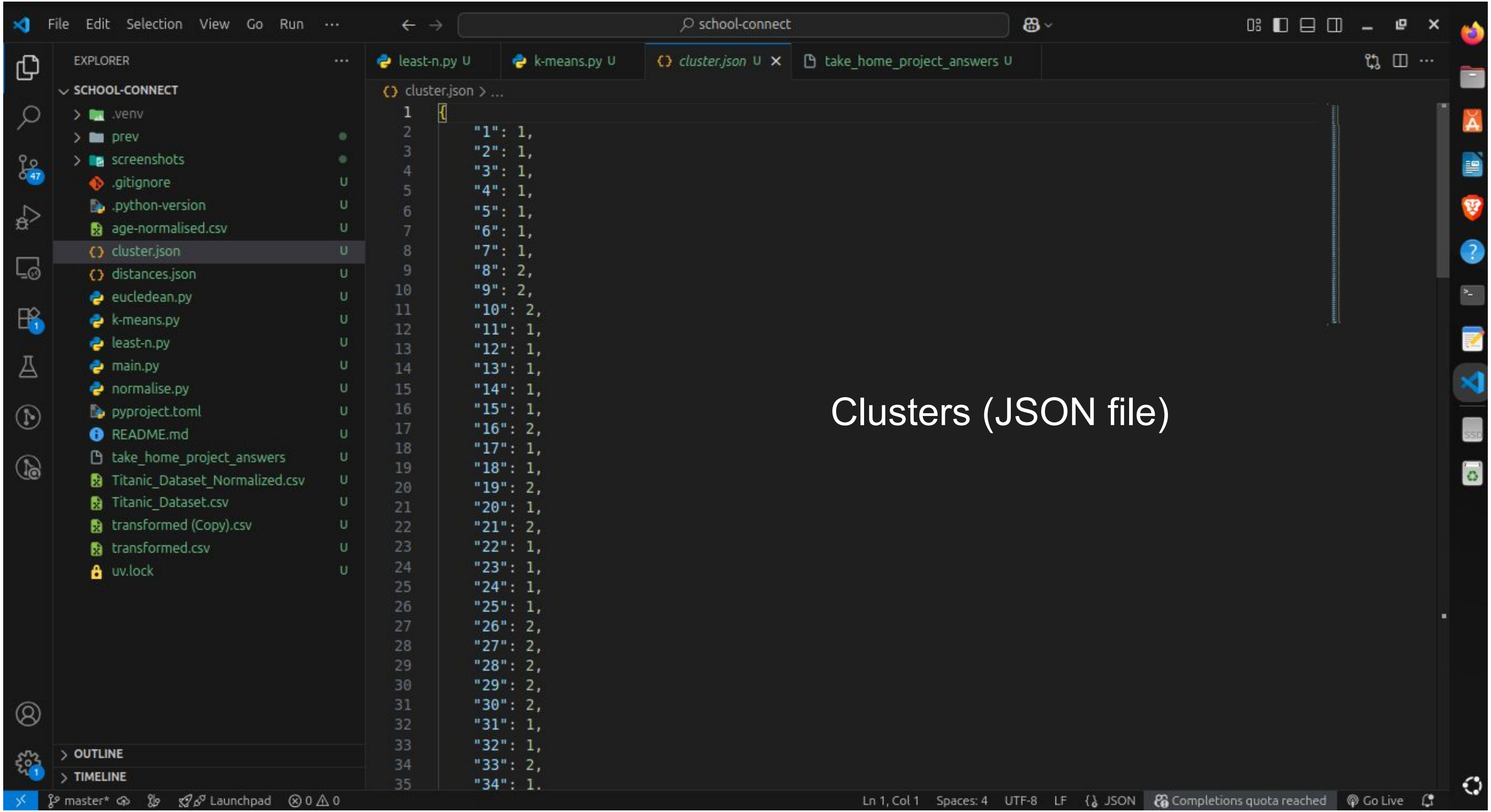


```python
import pandas as pd
from numpy.linalg import norm
import json
df = pd.read_csv("Titanic_Dataset.csv")
ids = df["PassengerID"]
df = df.drop(columns=["PassengerID"])

df["Gender"] = df["Gender"].map({"Male": 0, "Female": 1})
df["Embarked"] = df["Embarked"].map({"C": 0, "Q": 1, "S": 2})
```

```
╭─adidev at server in ~/Documents/school-connect on masterxxx using «.venv»
╰─± uv run k-means.py
Passenger 99 -> Cluster 2
Distance (Passenger 9 <-> Passenger 46) = 2.154
Cluster sizes:
 1    66
2    34
Name: count, dtype: int64
╭─adidev at server in ~/Documents/school-connect on masterxxx using «.venv»
╰─±
```

Result

Cluster Result



```
{
    "1": 1,
    "2": 1,
    "3": 1,
    "4": 1,
    "5": 1,
    "6": 1,
    "7": 1,
    "8": 2,
    "9": 2,
    "10": 2,
    "11": 1,
    "12": 1,
    "13": 1,
    "14": 1,
    "15": 1,
    "16": 2,
    "17": 1,
    "18": 1,
    "19": 2,
    "20": 1,
    "21": 2,
    "22": 1,
    "23": 1,
    "24": 1,
    "25": 1,
    "26": 2,
    "27": 2,
    "28": 2,
    "29": 2,
    "30": 2,
    "31": 1,
    "32": 1,
    "33": 2,
    "34": 1,
```

Clusters (JSON file)

Insights and Learnings

Trends/Patterns

- Females had much higher survival rates than males.
- 1st class passengers survived at higher rates compared to 2nd and 3rd class.
- Younger passengers (teens and children) had slightly higher survival chances than older groups.
- Passengers traveling **with others** had better survival rates compared to those alone.

Unique Insights

- Gender and class together formed the strongest predictors of survival.

- Traveling alone appeared to reduce survival chances -indicating the importance of family/social support.

- Embarkation port showed weaker relation with survival

Contribution of Tools

- In EDA section, the **distributions** chart helped a lot to unwind the details in the data

- In K-NN and K-Means, the lack of required tools helped me to create my **own data analysing programs**

- The techniques like **saving data in a JSON** format helped me to experiment with those values and optimise the analysis methods

Conclusion

- The project aimed to analyze survival patterns on the Titanic using data analysis tools

- Main findings: survival was influenced most by **gender, passenger class, and whether the passenger traveled alone**.

- **Social and economic** factors strongly affected survival in real-life disasters.
- Creating **custom tools** for analysis was a super interesting and rewarding task for me

References

- **Websites**
  - **Stackoverflow : https://stackoverflow.com/**
  - **Numpy Documentation : https://numpy.org/devdocs/user/**
  - **Pandas Documentation : https://pandas.pydata.org/docs/**
  - **Github : https://github.com/**

I created a github repository for this project
Project repository: https://github.com/adidev-c/school-connect-data-analysis