The Impact of Parental Employment on Student Achievement and financial purchase power

**Introduction**

Collecting the data, cleaning it and analysing the data to gain information for the benefit of a company is very important in today’s business.This project will be done from a newly appointed data scientists view. The goal here is to leverage data to find valuable insights that can drive company growth, particularly by improving products and potentially boosting sales.

The dataset the data scientist will be working with contains information on student demographics and family background. these backgrounds are social and financial information that can be used for a company benefit if mind right. The social aspect of a datasets are also important for the company in order to understand their customers.

tools are very important for data scientist in order to analyze the datasets.Exploratory Data Analysis (EDA) will be one of the main tools used in this assignment. This will include calculating, basic statistics and data visualisation during the process.

**Objective**

The primary goal of this project for the data scientist is to analyze the dataset on student demographics and family background to help the company in hand uncover secrets and patherns that can help with strategic decisions within the company.

**Scope**

 The company will understanding how demographic and social factors such as parental education level, family size, and living area relate to student performance or outcomes. Information like these can indirectly influence educational product development and customer engagement strategies.

**Data Collection**

For this assignment the writer will work with the "Student Performance" datasets, which contains information about students' demographics, family background, and academic performance. This dataset can potentially reveal patterns that relate to educational product improvements or marketing strategies targeting families that will benefit the company in long run.

pip install ucimlrepo

import pandas as pd

from ucimlrepo import fetch\_ucirepo

student\_performance = fetch\_ucirepo(id=320)

X = student\_performance.data.features

y = student\_performance.data.targets

print(student\_performance.metadata)

print(student\_performance.variables)

name role type demographic \

0 school Feature Categorical None

1 sex Feature Binary Sex

2 age Feature Integer Age

3 address Feature Categorical None

4 famsize Feature Categorical Other

5 Pstatus Feature Categorical Other

6 Medu Feature Integer Education Level

7 Fedu Feature Integer Education Level

8 Mjob Feature Categorical Occupation

9 Fjob Feature Categorical Occupation

10 reason Feature Categorical None

11 guardian Feature Categorical None

12 traveltime Feature Integer None

13 studytime Feature Integer None

14 failures Feature Integer None

15 schoolsup Feature Binary None

16 famsup Feature Binary None

17 paid Feature Binary None

18 activities Feature Binary None

19 nursery Feature Binary None

20 higher Feature Binary None

21 internet Feature Binary None

22 romantic Feature Binary None

23 famrel Feature Integer None

24 freetime Feature Integer None

25 goout Feature Integer None

26 Dalc Feature Integer None

27 Walc Feature Integer None

28 health Feature Integer None

29 absences Feature Integer None

30 G1 Target Categorical None

31 G2 Target Categorical None

32 G3 Target Integer None

description units missing\_values

0 student's school (binary: 'GP' - Gabriel Perei... None no

1 student's sex (binary: 'F' - female or 'M' - m... None no

2 student's age (numeric: from 15 to 22) None no

3 student's home address type (binary: 'U' - urb... None no

4 family size (binary: 'LE3' - less or equal to ... None no

5 parent's cohabitation status (binary: 'T' - li... None no

6 mother's education (numeric: 0 - none, 1 - pr... None no

7 father's education (numeric: 0 - none, 1 - pr... None no

8 mother's job (nominal: 'teacher', 'health' car... None no

9 father's job (nominal: 'teacher', 'health' car... None no

10 reason to choose this school (nominal: close t... None no

11 student's guardian (nominal: 'mother', 'father... None no

12 home to school travel time (numeric: 1 - <15 m... None no

13 weekly study time (numeric: 1 - <2 hours, 2 - ... None no

14 number of past class failures (numeric: n if 1... None no

15 extra educational support (binary: yes or no) None no

16 family educational support (binary: yes or no) None no

17 extra paid classes within the course subject (... None no

18 extra-curricular activities (binary: yes or no) None no

19 attended nursery school (binary: yes or no) None no

20 wants to take higher education (binary: yes or... None no

21 Internet access at home (binary: yes or no) None no

22 with a romantic relationship (binary: yes or no) None no

23 quality of family relationships (numeric: from... None no

24 free time after school (numeric: from 1 - very... None no

25 going out with friends (numeric: from 1 - very... None no

26 workday alcohol consumption (numeric: from 1 -... None no

27 weekend alcohol consumption (numeric: from 1 -... None no

28 current health status (numeric: from 1 - very ... None no

29 number of school absences (numeric: from 0 to 93) None no

30 first period grade (numeric: from 0 to 20) None no

31 second period grade (numeric: from 0 to 20) None no

32 final grade (numeric: from 0 to 20, output tar... None no

Load the dataset into a pandas DataFrame

df = pd.read\_csv('student-mat.csv', sep=';')

df.head(650)

A screenshot of a computer

Description automatically generated

Load the dataset into a pandas DataFrame

df = pd.read\_csv('student-por.csv', sep=';')

df.head(900)

A screenshot of a computer

Description automatically generated

**Cleaning the DataFrame Column**

The code below performs cleaning operations on a DataFrame column named absences in the datasets. It also handles missing values and removes duplicates.

column\_name = 'absences'

df[column\_name].fillna(value='Missing', inplace=True) # Replace NaN with 'Missing' or any other appropriate value

df.drop\_duplicates(subset=[column\_name], keep='first', inplace=True)

print(df.head(650))

school sex age address famsize Pstatus Medu Fedu Mjob Fjob \

0 GP F 18 U GT3 A 4 4 at\_home teacher

1 GP F 17 U GT3 T 1 1 at\_home other

2 GP F 15 U LE3 T 1 1 at\_home other

3 GP F 15 U GT3 T 4 2 health services

6 GP M 16 U LE3 T 2 2 other other

18 GP M 17 U GT3 T 3 2 services services

25 GP F 16 U GT3 T 2 2 services services

37 GP M 16 R GT3 A 4 4 other teacher

39 GP F 15 R GT3 T 2 2 at\_home other

40 GP F 16 U LE3 T 2 2 other other

46 GP F 16 U LE3 A 3 3 other services

74 GP F 16 U GT3 T 3 3 other services

89 GP M 16 U LE3 A 4 4 teacher health

103 GP F 15 U GT3 T 3 2 services other

118 GP M 17 R GT3 T 1 3 other other

183 GP F 17 U LE3 T 3 3 other other

198 GP F 17 U GT3 T 4 4 services teacher

205 GP F 17 U GT3 T 3 4 at\_home services

206 GP F 16 U GT3 A 3 1 services other

211 GP M 17 U LE3 T 4 4 services other

213 GP M 18 U GT3 T 2 2 services other

216 GP F 17 U GT3 T 4 3 other other

218 GP F 17 U GT3 T 2 3 at\_home other

260 GP F 18 U GT3 T 4 3 services other

262 GP M 18 R GT3 T 3 2 other other

276 GP F 18 R GT3 A 3 2 other services

280 GP M 17 U LE3 A 4 1 services other

281 GP M 17 U LE3 A 3 2 teacher services

289 GP M 18 U LE3 A 4 4 teacher teacher

290 GP M 18 U GT3 T 4 2 teacher other

307 GP M 19 U GT3 T 4 4 teacher services

315 GP F 19 R GT3 T 2 3 other other

320 GP F 17 U GT3 A 4 3 services services

379 MS F 17 R GT3 T 3 1 at\_home other

... famrel freetime goout Dalc Walc health absences G1 G2 G3

0 ... 4 3 4 1 1 3 6 5 6 6

1 ... 5 3 3 1 1 3 4 5 5 6

2 ... 4 3 2 2 3 3 10 7 8 10

3 ... 3 2 2 1 1 5 2 15 14 15

6 ... 4 4 4 1 1 3 0 12 12 11

18 ... 5 5 5 2 4 5 16 6 5 5

25 ... 1 2 2 1 3 5 14 6 9 8

37 ... 2 4 3 1 1 5 7 15 16 15

39 ... 4 3 1 1 1 2 8 14 13 13

40 ... 3 3 3 1 2 3 25 7 10 11

46 ... 2 3 5 1 4 3 12 11 12 11

74 ... 4 3 3 2 4 5 54 11 12 11

89 ... 4 1 3 3 5 5 18 8 6 7

103 ... 4 3 5 1 1 2 26 7 6 6

118 ... 5 2 4 1 4 5 20 9 7 8

183 ... 5 3 3 2 3 1 56 9 9 8

198 ... 4 2 4 2 3 2 24 18 18 18

205 ... 4 4 3 3 4 5 28 10 9 9

206 ... 2 3 3 2 2 4 5 7 7 7

211 ... 5 3 5 4 5 3 13 12 12 13

213 ... 4 4 4 2 4 5 15 6 7 8

216 ... 3 4 5 2 4 1 22 6 6 4

218 ... 3 3 3 1 4 3 3 7 7 8

260 ... 3 1 2 1 3 2 21 17 18 18

262 ... 5 3 2 1 1 3 1 13 12 12

276 ... 4 1 1 1 1 5 75 10 9 9

280 ... 4 5 4 2 4 5 30 8 8 8

281 ... 4 4 4 3 4 3 19 11 9 10

289 ... 5 4 3 1 1 2 9 15 13 15

290 ... 4 3 2 1 4 5 11 12 11 11

307 ... 4 3 4 1 1 4 38 8 9 8

315 ... 4 1 2 1 1 3 40 13 11 11

320 ... 5 2 2 1 2 5 23 13 13 13

379 ... 4 5 4 2 3 1 17 10 10 10

[34 rows x 33 columns]

**Filtering and Saving Student Data Based on Parents' Work Status**

now the writer loads the data sets from the 2 CSV files, filters the data based on the works of their parents, and then saves the filtered data to new CSV files.

**Load Data:** Reads student data from student-mat.csv and student-por.csv into DataFrames.

**Define Columns:** Specifies the columns needed for the assignment.

**Filter Data:** Filters students based on whether only the mother works, only the father works, or both parents work.

**Combine Results**: Merges the filtered results from both datasets, removes duplicates, and resets the index.

**Save to CSV:** Saves the combined results to separate CSV files for each category of parental work status.

import pandas as pd

df\_mat = pd.read\_csv('student-mat.csv', sep=';')

df\_por = pd.read\_csv('student-por.csv', sep=';')

columns\_needed = ['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'G1', 'G2', 'G3']

def filter\_students\_by\_parents\_work(df):

df = df[columns\_needed] # Keep only necessary columns

only\_mother\_works = df[(df['Mjob'] != 'at\_home') & (df['Fjob'] == 'at\_home')]

only\_father\_works = df[(df['Mjob'] == 'at\_home') & (df['Fjob'] != 'at\_home')]

both\_parents\_work = df[(df['Mjob'] != 'at\_home') & (df['Fjob'] != 'at\_home')]

return only\_mother\_works, only\_father\_works, both\_parents\_work

only\_mother\_works\_mat, only\_father\_works\_mat, both\_parents\_work\_mat = filter\_students\_by\_parents\_work(df\_mat)

only\_mother\_works\_por, only\_father\_works\_por, both\_parents\_work\_por = filter\_students\_by\_parents\_work(df\_por)

only\_mother\_works\_combined = pd.concat([only\_mother\_works\_mat, only\_mother\_works\_por]).drop\_duplicates().reset\_index(drop=True)

only\_father\_works\_combined = pd.concat([only\_father\_works\_mat, only\_father\_works\_por]).drop\_duplicates().reset\_index(drop=True)

both\_parents\_work\_combined = pd.concat([both\_parents\_work\_mat, both\_parents\_work\_por]).drop\_duplicates().reset\_index(drop=True)

only\_mother\_works\_combined.to\_csv('only\_mother\_works.csv', index=False)

only\_father\_works\_combined.to\_csv('only\_father\_works.csv', index=False)

both\_parents\_work\_combined.to\_csv('both\_parents\_work.csv', index=False)

print("CSV files have been saved successfully.")

**Filtering Student Data and Plotting Parental Education Levels**

The data scientist now perform data analysis on student datasets, filter the working status of there parents and visualise parental education levels.

1. **Load Data**: Reads student data from student-mat.csv and student-por.csv into DataFrames.
2. **Define Columns**: Specifies the columns needed for the analysis.
3. **Filter Data**: Filters students into three categories: only mother works, only father works, or both parents work.
4. **Combine Results**: Merges filtered results from both datasets, removes duplicates, and resets the index.
5. **Plot Data**: Visualizes the education levels of parents using bar plots for each category.

**Plotting Parental Education Levels**

1. **Setup Plot**: Creates a figure and sets its size.
2. **Convert Data Types**: Converts education levels to strings for proper labeling.
3. **Plot Data**: Uses Seaborn to create count plots, showing the distribution of education levels.
4. **Customize Plot**: Adds titles, labels, and legends for clarity.
5. **Display Plot**: Shows the plots for each category of parental work status.

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df\_mat = pd.read\_csv('student-mat.csv', sep=';')

df\_por = pd.read\_csv('student-por.csv', sep=';')

columns\_needed = ['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'G1', 'G2', 'G3']

def filter\_students\_by\_parents\_work(df):

df = df[columns\_needed] # Keep only necessary columns

only\_mother\_works = df[(df['Mjob'] != 'at\_home') & (df['Fjob'] == 'at\_home')]

only\_father\_works = df[(df['Mjob'] == 'at\_home') & (df['Fjob'] != 'at\_home')]

both\_parents\_work = df[(df['Mjob'] != 'at\_home') & (df['Fjob'] != 'at\_home')]

return only\_mother\_works, only\_father\_works, both\_parents\_work

only\_mother\_works\_mat, only\_father\_works\_mat, both\_parents\_work\_mat = filter\_students\_by\_parents\_work(df\_mat)

only\_mother\_works\_por, only\_father\_works\_por, both\_parents\_work\_por = filter\_students\_by\_parents\_work(df\_por)

only\_mother\_works\_combined = pd.concat([only\_mother\_works\_mat, only\_mother\_works\_por]).drop\_duplicates().reset\_index(drop=True)

only\_father\_works\_combined = pd.concat([only\_father\_works\_mat, only\_father\_works\_por]).drop\_duplicates().reset\_index(drop=True)

both\_parents\_work\_combined = pd.concat([both\_parents\_work\_mat, both\_parents\_work\_por]).drop\_duplicates().reset\_index(drop=True)

def plot\_parental\_education(df, title):

plt.figure(figsize=(10, 6))

df['Medu'] = df['Medu'].astype(str)

df['Fedu'] = df['Fedu'].astype(str)

sns.countplot(x='Medu', data=df, hue='Fedu', palette='pastel')

plt.title(title)

plt.xlabel("Mother's Education Level")

plt.ylabel('Count')

plt.legend(title="Father's Education Level", loc='upper right')

plt.show()

plot\_parental\_education(only\_mother\_works\_combined, 'Parental Education Levels (Only Mother Works)')

plot\_parental\_education(only\_father\_works\_combined, 'Parental Education Levels (Only Father Works)')

plot\_parental\_education(both\_parents\_work\_combined, 'Parental Education Levels (Both Parents Work)')

A graph of a parent's education

Description automatically generated

A graph of different colored bars

Description automatically generated

A graph of different colored columns

Description automatically generated

**Visualizing the Distribution of Students Based on Parental Working Status**

Now the data scientist uses the code below to visualise the datasets of the students on whether the mother work ,the father works orboth the parents work and counts them. After doing that, he creates a visual graph.

1. **Count Students**: Counts the number of students in each category: only mother working, only father working, and both parents working.
2. **Define Labels and Colors**: Sets labels for the categories and chooses colors for the bar segments.
3. **Create Bar Plot**: Plots a stacked bar chart to display the distribution of students across the three categories.
4. **Customize Plot**: Adds titles, labels, and adjusts font sizes for better readability.
5. **text Plot**: Adds text on each bar segment to show the count of students.
6. **Display Plot**: Shows the final plot.

counts\_mother = len(only\_mother\_works\_combined)

counts\_father = len(only\_father\_works\_combined)

counts\_both = len(both\_parents\_work\_combined)

labels = ['Only Mother Working', 'Only Father Working', 'Both Parents Working']

colors = ['#66c2a5', '#fc8d62', '#8da0cb']

plt.figure(figsize=(10, 6))

p1 = plt.bar(labels[0], counts\_mother, color=colors[0])

p2 = plt.bar(labels[1], counts\_father, bottom=counts\_mother, color=colors[1])

p3 = plt.bar(labels[2], counts\_both, bottom=counts\_mother + counts\_father, color=colors[2])

plt.title('Distribution of Students Based on Parental Working Status', fontsize=18, fontweight='bold')

plt.xlabel('Parental Working Status', fontsize=14)

plt.ylabel('Number of Students', fontsize=14)

plt.xticks(fontsize=12)

plt.yticks(fontsize=12)

plt.legend((p1, p2, p3), ('Only Mother Working', 'Only Father Working', 'Both Parents Working'), loc='upper right', fontsize=12)

plt.grid(False)

for rect in p1 + p2 + p3:

height = rect.get\_height()

plt.annotate('{}'.format(height),

xy=(rect.get\_x() + rect.get\_width() / 2, height),

xytext=(0, 3), # 3 points vertical offset

textcoords="offset points",

ha='center', va='bottom', fontsize=10)

plt.tight\_layout()

plt.show()

A graph with text on it

Description automatically generated

**Analyzing Basic Statistics of Filtered Student Data**

The code below calculates and prints basic statistical summaries for datasets after being filtered by parental working status. The code is useful for having an overview of the data, including metrics such as count, mean, standard deviation, minimum, maximum, and quartiles for each numeric column. The output afterwards consists of printed summaries for each dataset, showing basic statistics like the number of samples, mean values, and distribution ranges for numeric data columns. All these combine can be used to gain a more accurate and statistical view of the data sets.

df\_list = [only\_mother\_works\_combined, only\_father\_works\_combined, both\_parents\_work\_combined]

for i, df in enumerate(df\_list):

print(f"Dataset {i+1} - Number of samples: {len(df)}")

print(df.describe())

print("\n")

Dataset 1 - Number of samples: 37

age Medu Fedu G1 G2 G3

count 37.000000 37.000000 37.000000 37.000000 37.000000 37.000000

mean 17.000000 2.621622 2.162162 9.972973 9.891892 10.378378

std 1.333333 1.209944 1.014164 2.733399 3.454335 3.514663

min 15.000000 0.000000 1.000000 5.000000 0.000000 0.000000

25% 16.000000 2.000000 1.000000 8.000000 9.000000 9.000000

50% 17.000000 3.000000 2.000000 10.000000 10.000000 11.000000

75% 18.000000 4.000000 3.000000 12.000000 12.000000 13.000000

max 19.000000 4.000000 4.000000 17.000000 16.000000 16.000000

Dataset 2 - Number of samples: 169

age Medu Fedu G1 G2 G3

count 169.000000 169.000000 169.000000 169.000000 169.000000 169.000000

mean 16.934911 1.710059 1.976331 10.213018 10.165680 10.248521

std 1.268398 0.854933 1.057584 2.872625 3.137106 3.621663

min 15.000000 0.000000 0.000000 0.000000 0.000000 0.000000

25% 16.000000 1.000000 1.000000 8.000000 9.000000 9.000000

50% 17.000000 2.000000 2.000000 10.000000 10.000000 10.000000

75% 18.000000 2.000000 3.000000 12.000000 12.000000 12.000000

max 21.000000 4.000000 4.000000 19.000000 17.000000 18.000000

Dataset 3 - Number of samples: 809

age Medu Fedu G1 G2 G3

count 809.000000 809.000000 809.000000 809.000000 809.000000 809.000000

mean 16.658838 2.824475 2.505562 11.452410 11.512979 11.595797

std 1.233725 1.064199 1.091271 2.954778 3.242683 3.874713

min 15.000000 0.000000 0.000000 3.000000 0.000000 0.000000

25% 16.000000 2.000000 2.000000 9.000000 9.000000 10.000000

50% 17.000000 3.000000 2.000000 11.000000 12.000000 12.000000

75% 18.000000 4.000000 3.000000 14.000000 14.000000 14.000000

max 22.000000 4.000000 4.000000 19.000000 19.000000 20.000000

This code creates boxplots to visualize the distribution of student grades (G1, G2, G3) for different parental working statuses. The main use for this code isto visualize and compare the distributions of students grades across different parental working statuses. Boxplots is also used as a tool here to provide a clear summary of the central tendency, spread, and potential outliers in the data. The output is a boxplot that shows the distribution of grades for students whose only mother works, only father works or both parents work.

1. **Combine Datasets**: Combines the pre-filtered datasets into a single DataFrame.
2. **Melt Data**: Reshapes the data to a long format suitable for Seaborn's boxplot function.
3. **Create Boxplot**: Plots boxplots to show the distribution of grades across different categories.
4. **Customize Plot**: Adds titles, labels, and legend to the plot.

plt.figure(figsize=(12, 8))

sns.boxplot(x='variable', y='value', data=pd.melt(pd.concat(df\_list)[['G1', 'G2', 'G3']]), hue='variable', palette='Set3')

plt.title('Boxplot of Grades by Parental Working Status', fontsize=16)

plt.xlabel('Grades', fontsize=14)

plt.ylabel('Grade Value', fontsize=14)

plt.legend(title='Grades', fontsize=12)

plt.show()

A chart with a box and a few boxes

Description automatically generated with medium confidence

**Chi-Square Test for Parental Working Status Distribution**

Next step is Chi-square test to analyze whether there is a significant difference in the data sets of students based on parental working status. Thecode is used to test whether the observed distribution of students based on parental working status is deferent significantly from what might be expected by chance. It is a statistical test that helps determine if there is an association between two catagories of variables in here.The output consists of the chi-square statistic and the p-value. The chi-square statistic indicates the extent of difference between the observed and expected frequencies, while the p-value indicates the significance of the result..The steps below show how this works.

1 **Count Students**: Calculates the number of students in each category: only mother working, only father working, and both parents working.

2 **Create Contingency Table**: Constructs a contingency table with the counts.

3 **Perform Chi-Square Test**: Uses the chi-square test to determine if there is a significant difference in the distribution.

4 **Print Results**: Outputs the chi-square statistic and the p-value.

import numpy as np

from scipy.stats import chi2\_contingency

counts\_mother = len(only\_mother\_works\_combined)

counts\_father = len(only\_father\_works\_combined)

counts\_both = len(both\_parents\_work\_combined)

observed = np.array([[counts\_mother, counts\_father, counts\_both]])

chi2\_stat, p\_val, \_, \_ = chi2\_contingency(observed)

print(f"Chi-square statistic: {chi2\_stat}")

print(f"P-value: {p\_val}")

Result: Chi-square statistic: 0.0

P-value: 1.0

Conclusion

In this assignment the data scientist did a big analysis of student datasets collected from internet and then sorted it by how parents work using Python libraries like pandas, seaborn, and scipy. First, the data was cleaned to make sure it's good, fixing missing stuff and taking out copies in the CSV files 'student-mat.csv' and 'student-por.csv'. Then, after doing that he looked at things like how much education parents have and how kids grades (like G1, G2, G3) change based on if mom works, dad works, or both work. he used bar graphs and box graphs to see these things better. He also used a test called chi-square to see if there's a real connection between how parents work and where kids end up. What he found shows us how family life can really affect how kids do in school. This study helps the company see more about how data can explain things in education and gives ideas for helping kids in different family situations do better in school. Then this show how it is important for the company to not just analyze the financial aspects, but also the social aspect of these customers in order to understand them and know the companies targets by figuring out which of these families make the most money, and also which of them are comfortable spending their money on their children.

DATASET LINK:

<https://archive.ics.uci.edu/dataset/320/student+performance>

JUPYTERS NOTEBOOK LINK:

<http://localhost:8901/lab/tree/data%20mining.ipynb>



Assessment Submission Form

|  |  |
| --- | --- |
| **Student Number**  (If this is group work, please  include the student numbers of all group participants) | GH1024093 |
| **Assessment Title** | The Impact of Parental Employment on Student Achievement and financial purchase power |
| **Module Code** | B141 |
| **Module Title** | Data Mining |
| **Module Tutor** | Mahmoudreza Babaei |
| **Date Submitted** | 03/07/2024 |

**Declaration of Authorship**

I declare that all material in this assessment is my own work except where there is clear acknowledgement and appropriate reference to the work of others.

I fully understand that the unacknowledged inclusion of another person’s writings or ideas or works in this work may be considered plagiarism and that, should a formal investigation process confirms the allegation, I would be subject to the penalties associated with plagiarism, as per GISMA Business School, University of Applied Sciences’ regulations for academic misconduct.

Signed………………Ali Jawed Delawari……………. Date ………03/07/2024………………