
Introduction to Intelligent Systems

Assignment 1

Report

Aditya Prasad
2022036
BTech 2026
CSAI

Section A (Theoretical)

Q1

(a) Rules of inference

Predicate Rules:

1) Universal Instantiation: It is used to derive specific instances of universally quantified statements.

$$\forall x (\text{dog}(x) \rightarrow \text{bark}(x))$$

From the above we can infer that $\text{bark}(x)$ means X barks since X is a dog.

2) Universal Generalization: It is used to ensure if a statement is true for a specific domain, then it is true for all elements in that domain.

It can be represented as $A(x) \vdash \forall x A(x)$

Eg: If we know that a specific cat is a mammal, we can use universal generalization to conclude that all cats are mammals.

3) Existential Generalization: It states that if a statement is true for some particular element in a domain, then we can give a conclusion that there exists at least one element in the domain for which the statement is true.

It can be represented as $T(a) \vee \exists x T(x)$

Here a is any specific element in that domain.

Eg: If we are given a particular player, who is playing a match, we can say that one of the players in the team is playing the match.

4) Existential Instantiation: It allows you to derive an instance of an existentially quantified statement.

$\exists x (\text{dog}(x) \wedge \text{bark}(x))$

Eg: We can infer $\text{bark}(x)$ by introducing a new variable y and substituting x for y since x is a dog that barks.

Propositional Rules:

1) Modus Ponens: It is used to derive new statements from existing ones.

If A implies B and A is true, then B must be true as well.

It is represented as $(A \rightarrow B) \wedge A \rightarrow B$

Eg: If it's raining, the ground is wet. Thus if it is raining a conclusion can be made that the ground is wet.

2) Modus Tollens: It is used to derive new statements from existing ones.

If A implies B and A is false, then B must be false as well.

It is represented as $(A \rightarrow B) \wedge \sim B \rightarrow \sim A$

Eg: If it's raining, the ground is wet. Thus if it is given that the ground is not wet we can conclude that it is not raining.

3)Law of syllogism: It is used to derive new statements from existing ones.

If A implies B and B implies C, then A implies C.

Symbolically it can be represented as $(A \rightarrow B) \wedge (B \rightarrow C) \rightarrow (A \rightarrow C)$

Eg: If it's raining, the ground is wet and if the ground is wet the shoes are dirty. Now, if it is given that it's raining, we can say that the shoes are dirty.

4)Addition: It is used to give us or statements for two conditions.

If P is true, then P or Q is true.

Symbolically it can be represented as $A \Rightarrow A \vee B$

Eg: I will eat breakfast if A implies I will eat breakfast, or I will sleep.

5)Simplification: It gives a single statement from two statements as both are true.

If A and B are true, then A is true.

Symbolically it can be represented as $(A \vee B) \Rightarrow A$

Eg: I will eat breakfast and read a book. Thus if it rains I will read a book.

Q2)

Machine learning plays a vital role in automating, decision-making and pattern recognition tasks.

i) An applicant's credit card application will either be approved or rejected by a credit card provider using machine learning (ML). The application for a credit card can be automatically approved or rejected using machine

learning (ML). It has the capacity to learn various patterns and facts to draw the conclusion of a qualified candidate. However, using an ML raises ethical concerns because it can be biased depending on the training dataset used to train it. If the data is biased, the ML model will start to discriminate against some groups, which will result in unfair decisions. Because of these drawbacks, we should not use ML.

ii)A hospital hopes to employ machine learning to create individualized treatment regimens by forecasting patient outcomes. This can help the hospital deliver reports at a much faster rate, and the machine learning model can learn to predict which treatments are likely to be successful for a particular patient. However, the ML model may see some symptoms and would generalize the cure for every patient, though it would speed up the work it may also have negative effects. Additionally, patient data needs to be protected and should be careful with security so it doesn't get into the wrong hands if these things are monitored ML would be a great help in hospitals and develop personalized treatment plans.

iii)The school hopes to employ machine learning to monitor pupils' academic progress and offer tailored suggestions for development. Students that require academic assistance can benefit from the use of ML, which can also be customized and utilized to train students to excel in a particular field. Additionally, ML can be used in this sector to identify the various areas in which a particular student is poor so that teachers can focus more of their efforts on that kid in that area. Thus machine learning would be a beneficial step to secure the future of the students in school.

iv)ML is being used by a law enforcement agency to find prospective criminal offenders. By training ML models on crime datasets, law enforcement can utilize it to help identify prospective suspects. By doing this, the investigative team can reduce the number of probable suspects, which can shorten the time it takes to solve a case. However, this could lead to bias on the part of the ML due to the training given to it on the crime data set. ML should not be utilized in this situation since it can be biased

towards a particular group of people based on the training given to it on the crime data set.

Q3)

The Turing Test is an evaluation that is frequently used to gauge a machine's level of intelligence. A machine's ability to exhibit behavior that is identical to human behavior can be tested using the Turing test. A sequence of questions are posed to both respondents during the test by the human questioner. The questioner tries to determine which terminal is operated by a human respondent and which terminal is operated by a computer after the allotted amount of time.

When this test was run on ChatGPT, the following observations were made:

- The jobs that require learning new things were handled by Chat GPT with excellent proficiency. It had the capacity to recognise patterns and, using that knowledge, create an algorithm that was effective for situations of a similar nature.
- Since ChatGPT was unable to fully comprehend the context of human talks, it was unable to create a long-term thread of dialogues.
- However, it was unable to perform as expected when it came to tasks that required understanding and effectively utilizing complicated emotional or social skills.
- Additionally, it fails to provide an adequate response to issues requiring original thought and creative problem-solving techniques.

Section B (Code Implementation)

Q1) EDA on the Titanic Dataset

i) The first line prints the shape of the dataset, which is the number of rows and columns in the dataset.

The second line prints the first ten rows of the dataset using the `.head()` method.

The third line prints the last ten rows of the dataset using the `.tail()` method.

Overall, this code is a basic exploration of the Titanic dataset, providing information about the size of the dataset and showing some of the data contained within it.

```
#!/usr/bin/env python
# Print shape of the dataset
print("Shape of Titanic dataset: ", titanic_df.shape)

# Show first ten rows
print("First ten rows:")
print(titanic_df.head(10))
print()

# Show last ten rows
print("Last ten rows:")
print(titanic_df.tail(10))
print()
```

Shape of Titanic dataset: (891, 12)

First ten rows:

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
2	3	1	3	
3	4	1	1	
4	5	0	3	
5	6	0	3	
6	7	0	1	
7	8	0	3	
8	9	1	3	
9	10	1	2	

	Name	Sex	Age	SibSp	\
0	Braund, Mr. Owen Harris	male	22.0	1	
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	
2	Heikkinen, Miss. Laina	female	26.0	0	
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	
4	Allen, Mr. William Henry	male	35.0	0	
5	Moran, Mr. James	male	NaN	0	
6	McCarthy, Mr. Timothy J	male	54.0	0	
7	Palsson, Master. Gosta Leonard	male	2.0	3	
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	
9	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	

	Parch	Ticket	Fare	Cabin	Embarked
0	0	A/5 21171	7.2500	NaN	S
1	0	PC 17599	71.2833	C85	C
2	0	STON/O2. 3101282	7.9250	NaN	S
3	0	113803	53.1000	C123	S
4	0	373450	8.0500	NaN	S
5	0	330877	8.4583	NaN	Q
6	0	17463	51.8625	E46	S
7	1	349909	21.0750	NaN	S
8	2	347742	11.1333	NaN	S
9	0	237736	30.0708	NaN	C

```

Last ten rows:
  PassengerId  Survived  Pclass
881         882         0      3
882         883         0      3
883         884         0      2
884         885         0      3
885         886         0      3
886         887         0      2
887         888         1      1
888         889         0      3
889         890         1      1
890         891         0      3

      Name \
881      Markun, Mr. Johann
882  Dahlberg, Miss. Gerda Ulrika
883  Banfield, Mr. Frederick James
884    Sutehall, Mr. Henry Jr
885  Rice, Mrs. William (Margaret Norton)
886    Montvila, Rev. Juozas
887    Graham, Miss. Margaret Edith
888  Johnston, Miss. Catherine Helen "Carrie"
889    Behr, Mr. Karl Howell
890    Dooley, Mr. Patrick

  Sex  Age  SibSp  Parch  Ticket   Fare Cabin Embarked
881  male  33.0    0    0   349257   7.8958   NaN        S
882  female  22.0    0    0    7552  10.5167   NaN        S
883  male  28.0    0    0  C.A./SOTON 34068  10.5000   NaN        S
884  male  25.0    0    0  SOTON/OQ 392076   7.0500   NaN        S
885  female  39.0    0    5   382652  29.1250   NaN        Q
886  male  27.0    0    0   211536  13.0000   NaN        S
887  female  19.0    0    0   112053  30.0000   B42        S
888  female   NaN    1    2    W./C. 6607  23.4500   NaN        S
889  male  26.0    0    0   111369  30.0000  C148        C
890  male  32.0    0    0   370376   7.7500   NaN        Q

```

ii) The first line creates a variable called "null_values" and assigns it the result of calling the .isnull() method on the "titanic_df" dataset, which returns a boolean DataFrame indicating whether each element of the dataset is null or not. The .sum() method is then called on this boolean DataFrame to count the number of null values in each column.

The second line prints a message indicating that it will display the count of null values in each column of the dataset. The null value counts are then printed to the console using the print() function.

Overall, this code is a basic check for missing or null values in the "titanic_df" dataset, which is an important step in any data analysis or preprocessing task.

```

#(ii)

# Check for null values
null_values = titanic_df.isnull().sum()

# Print the null values count
print('Null values count in each column:')
print(null_values)
print()

```

```
Null values count in each column:
PassengerId      0
Survived          0
Pclass           0
Name             0
Sex              0
Age             177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin           687
Embarked         2
dtype: int64
```

The first line sets the plotting style to 'whitegrid' using the `sns.set_style()` function from the Seaborn library.

A for loop is then used to iterate over all columns in the "titanic_df" dataset. For each column, a new figure is created using `plt.figure()`, and a countplot is generated using `sns.countplot()`. The x parameter of the countplot is set to the current column being iterated over, and the data parameter is set to the "titanic_df" dataset.

The `xlabel()`, `ylabel()`, and `title()` functions are then used to label the axes and title of each countplot. The `xlabel()` is set to the current column being iterated over, while the `ylabel()` is set to 'Count', and the `title()` is set to 'Countplot of [column name] in Titanic Training Dataset'.

Finally, `plt.show()` is called to display all of the count plots.

Overall, this code is a basic way to visualize the distribution of values for each column in the "titanic_df" dataset using countplots, which can be useful for identifying patterns and relationships in the data.

```
# Plot the countplot for all columns
sns.set_style('whitegrid')
for col in titanic_df.columns:
    plt.figure()
    sns.countplot(x=col, data=titanic_df)
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.title('Countplot of ' + col + ' in Titanic Training Dataset')

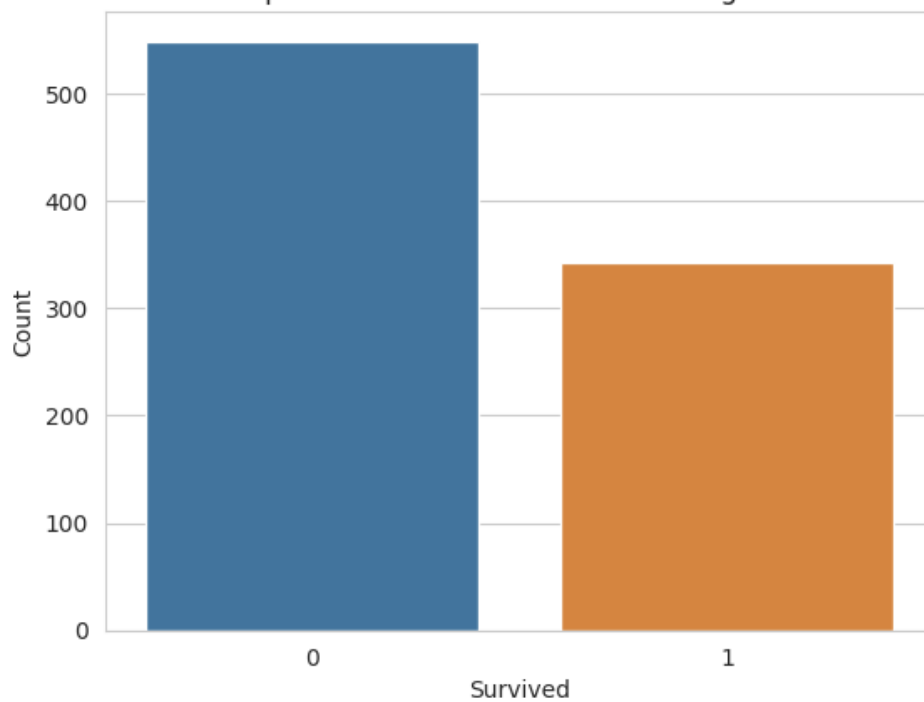
# Show the plots
plt.show()
```



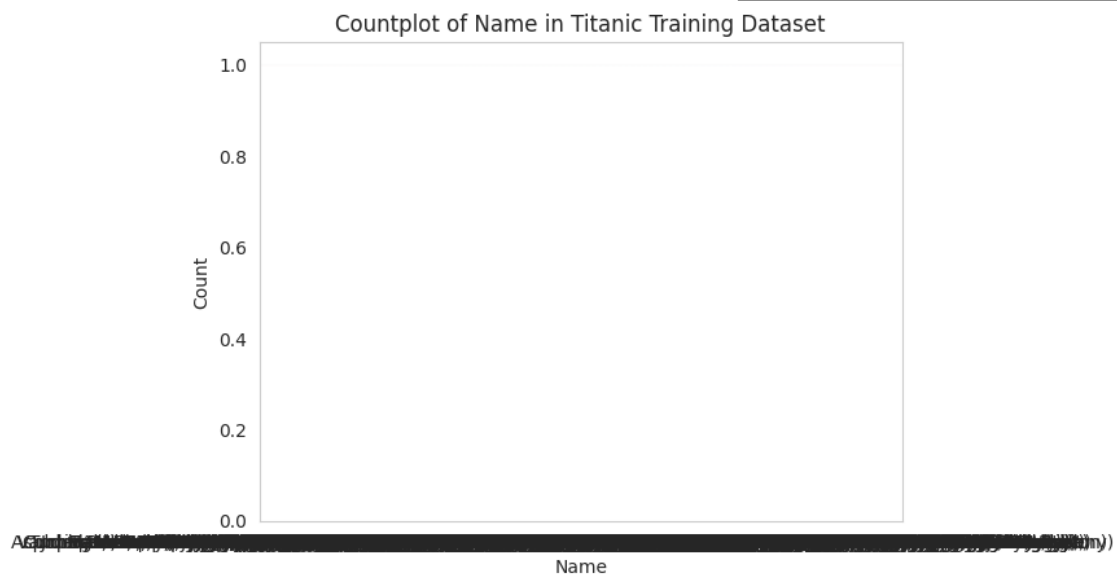
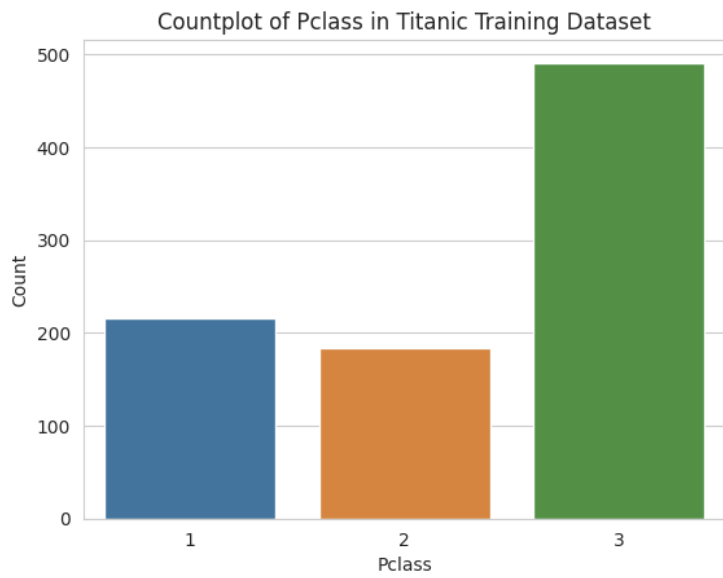

Countplot of PassengerId in Titanic Training Dataset



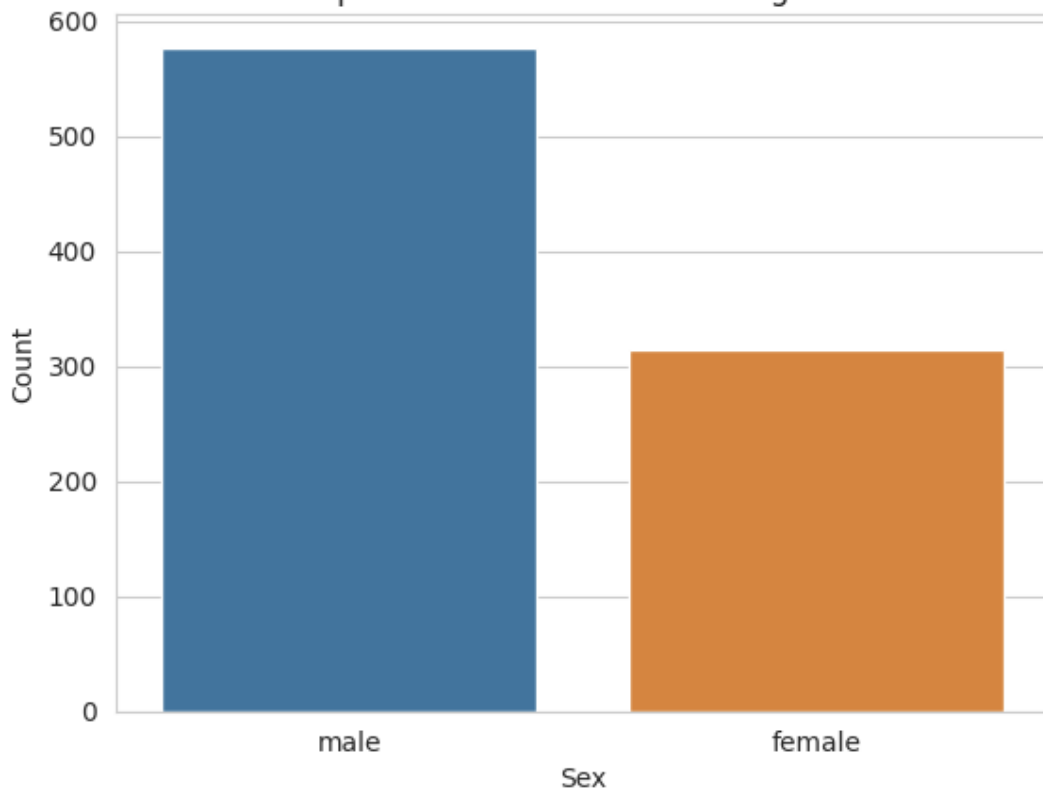
Countplot of Survived in Titanic Training Dataset



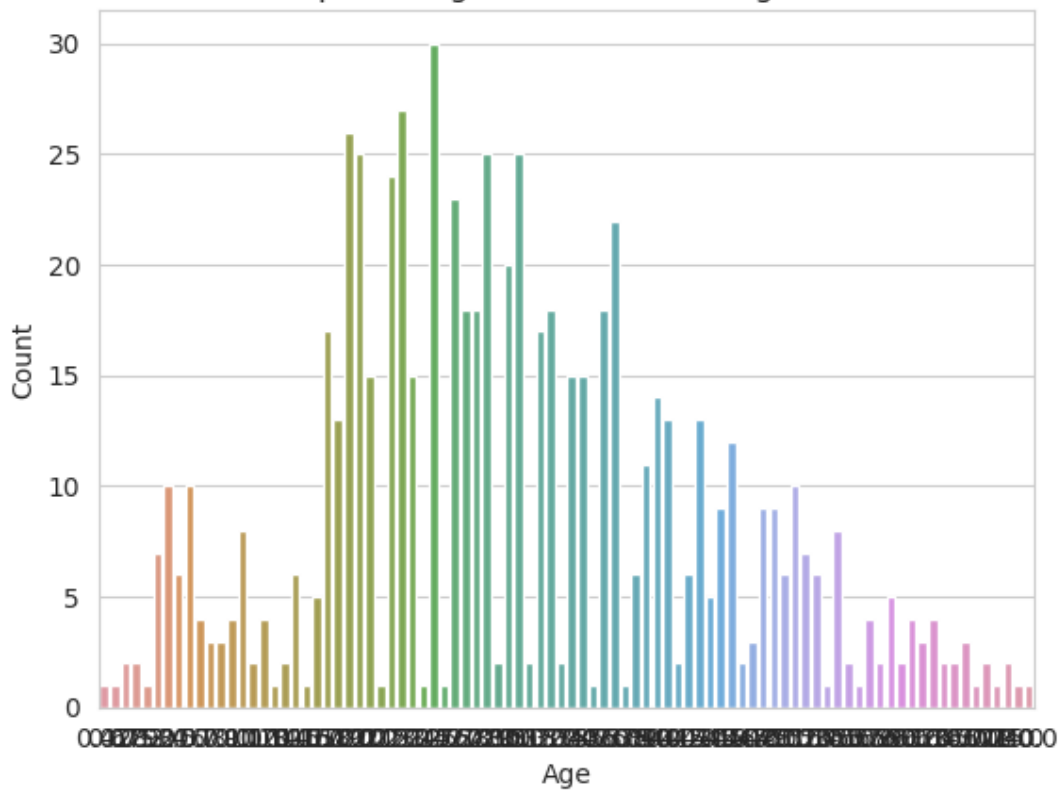
Countplot of Pclass in Titanic Training Dataset



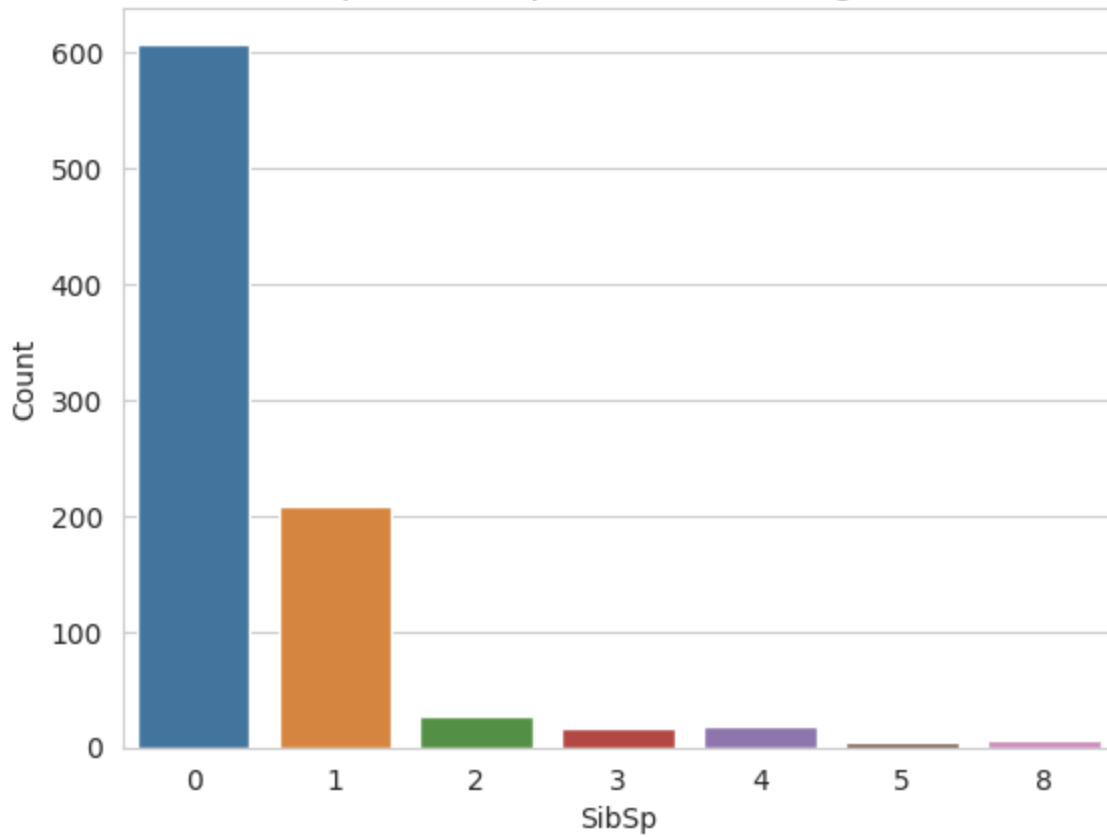
Countplot of Sex in Titanic Training Dataset



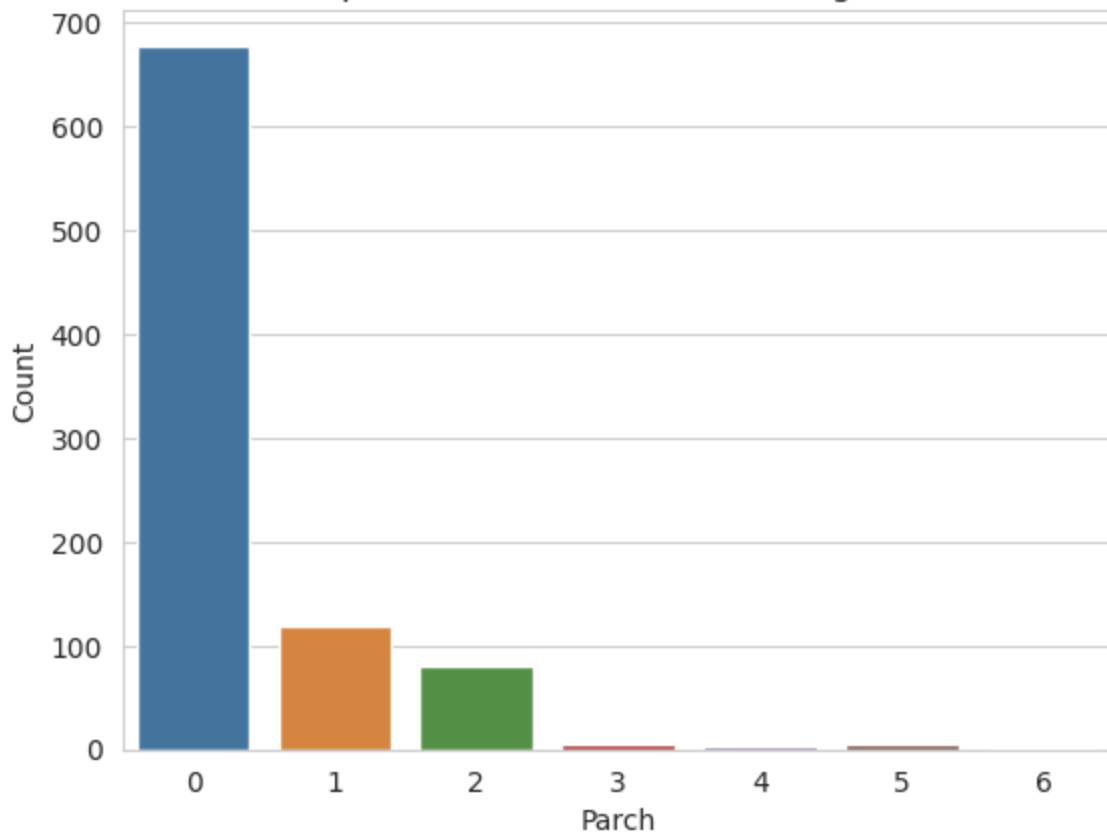
Countplot of Age in Titanic Training Dataset



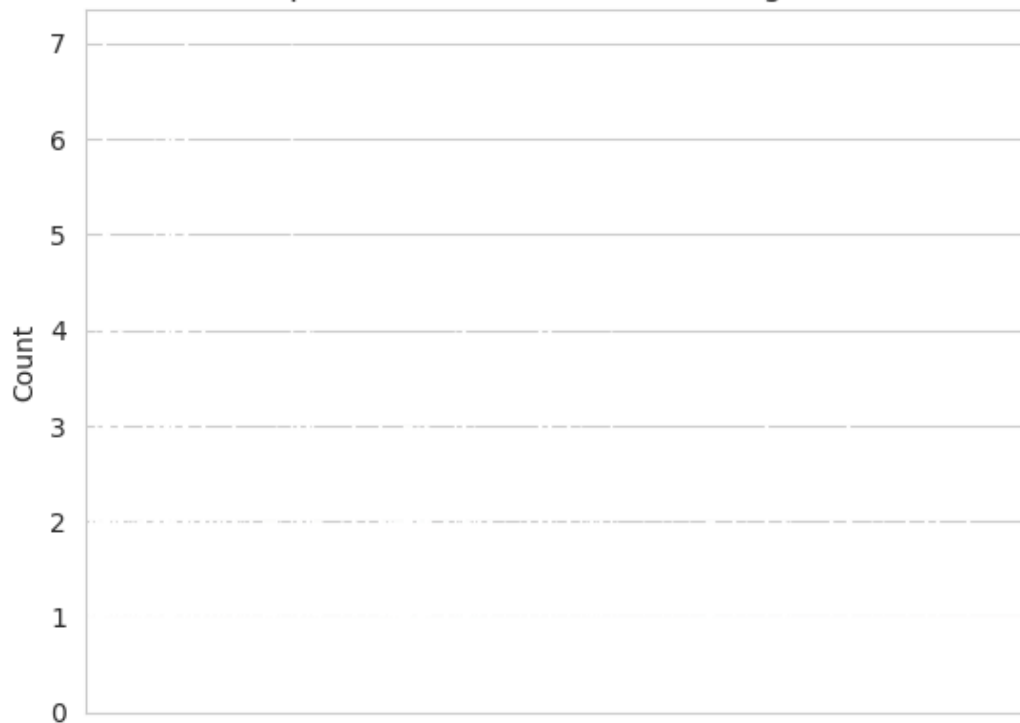
Countplot of SibSp in Titanic Training Dataset



Countplot of Parch in Titanic Training Dataset

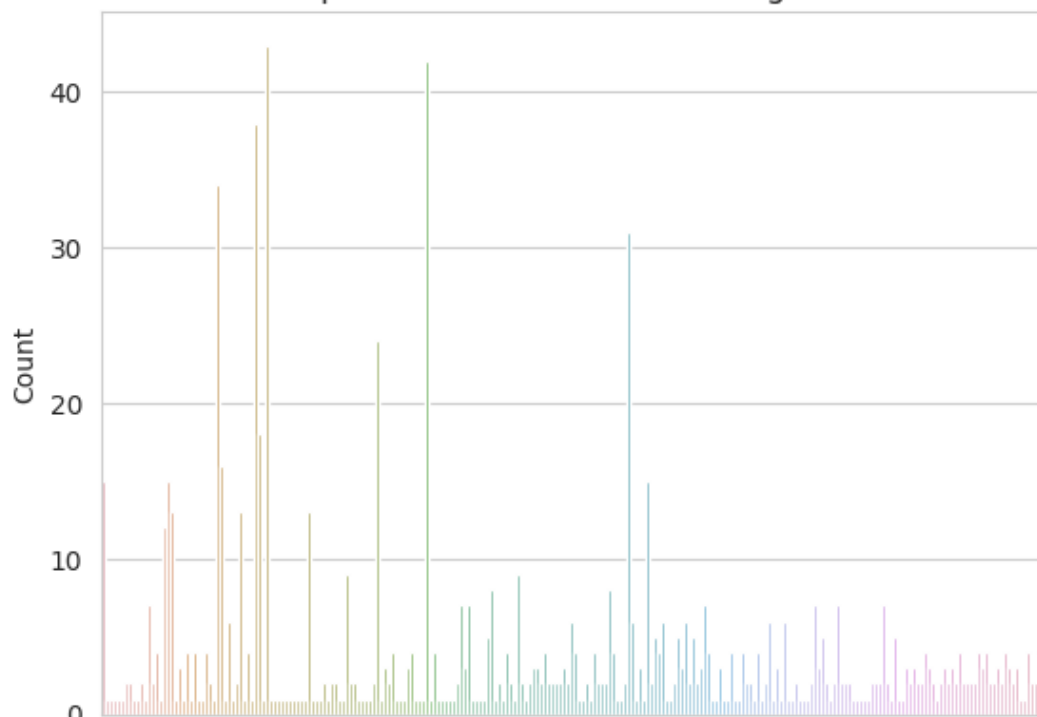


Countplot of Ticket in Titanic Training Dataset

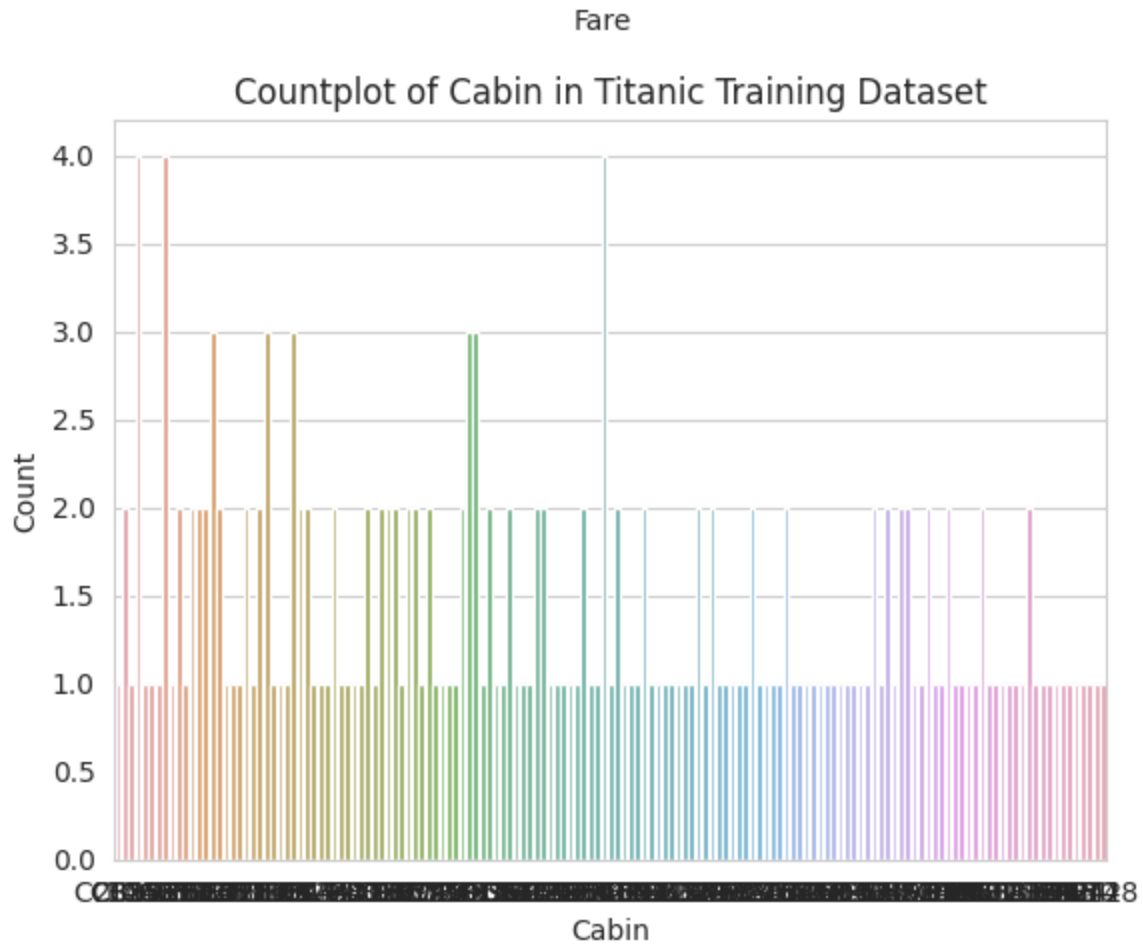


STON 46078

Countplot of Fare in Titanic Training Dataset



46078 10082

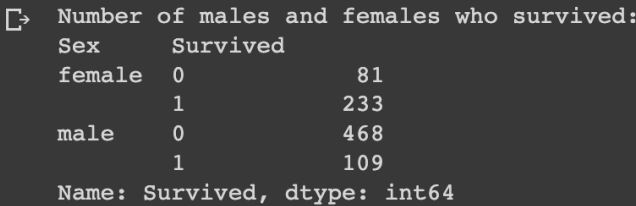


iv) The first line of code calls the `.groupby()` method on the "titanic_df" dataset, passing in a list of the columns to group by. The `.Survived.count()` method is then called on the resulting grouped object to count the number of occurrences of each combination of values. The second line of code prints a message indicating that the result will show the number of males and females who survived, and then prints the 'gender_survival' variable to the console. Overall, this code is a basic way to group and summarize the "titanic_df" dataset by the 'Sex' and 'Survived' columns to obtain information about the number of males and females who survived the sinking of the Titanic.

```
#(iv) _____

# Group by 'Sex' column and count 'Survived' column
gender_survival = titanic_df.groupby(['Sex', 'Survived']).Survived.count()

# Print the result
print('Number of males and females who survived:')
print(gender_survival)
print()
```



Number of males and females who survived:

Sex	Survived	
female	0	81
	1	233
male	0	468
	1	109

Name: Survived, dtype: int64

v) The first line of code calls the `sns.countplot()` function from the Seaborn library, passing in the 'Pclass' column as the x parameter, and the 'Survived' column as the hue parameter. The data parameter is set to the "titanic_df" dataset. The second line of code sets the `xlabel()`, `ylabel()`, and `title()` of the plot using the `plt.xlabel()`, `plt.ylabel()`, and `plt.title()` functions, respectively. Finally, `plt.show()` is called to display the countplot. Overall, this code is a basic way to visualize the distribution of survivors and non-survivors across the different passenger classes on the Titanic using a countplot. The hue parameter allows us to compare the survival rate for each passenger class.

```

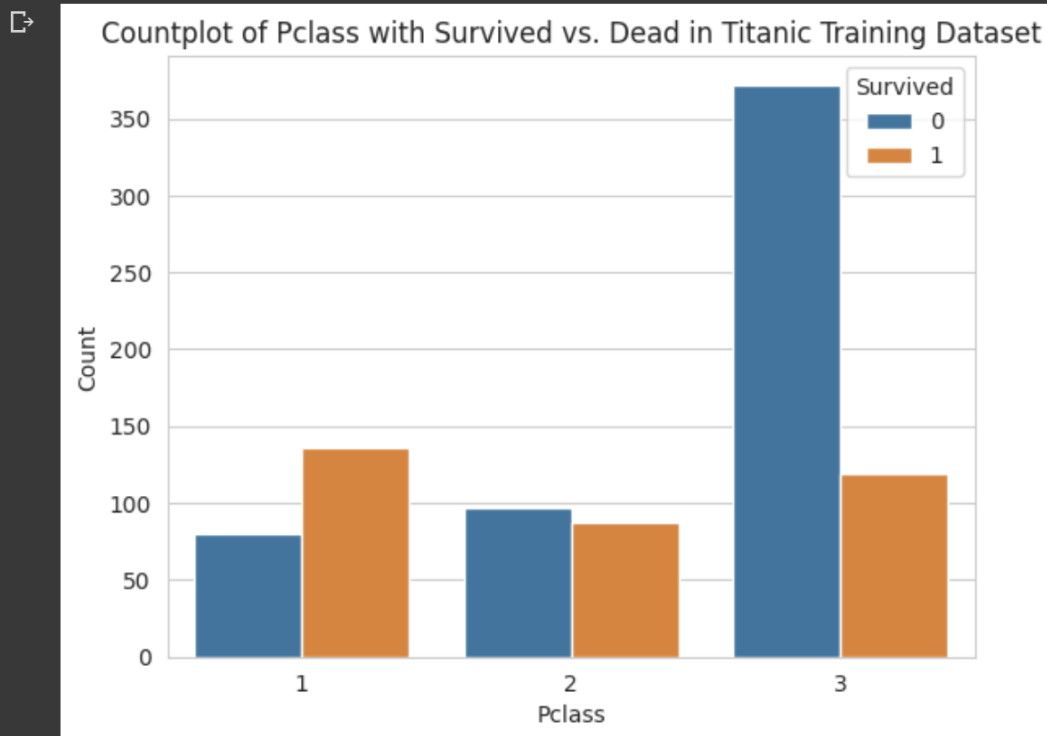
#(v) _____

# Plot the countplot for 'Pclass' with hue as 'Survived'
sns.countplot(x='Pclass', hue='Survived', data=titanic_df)

# Set labels and title
plt.xlabel('Pclass')
plt.ylabel('Count')
plt.title('Countplot of Pclass with Survived vs. Dead in Titanic Training Dataset')

# Show the plot
plt.show()
print()

```



vi) The first line of code filters the "titanic_df" dataset to select only rows where the 'Survived' column is equal to 1. This is accomplished using boolean indexing. The resulting filtered dataset is stored in a new variable called 'survived_df'.

The next three lines of code use various pandas methods to calculate the oldest, youngest, and average age of the passengers who survived in the 'survived_df' dataset. The .max() method is used to find the maximum value of the 'Age' column, the .min() method is used to find the minimum value, and the .mean() method is used to calculate the average.

Finally, the last four lines of code print out the results in a human-readable format using the .format() method to insert the calculated values into the printed string.

Overall, this code is a simple way to filter and summarize the "titanic_df" dataset to obtain information about the ages of passengers who survived the sinking of the Titanic.


```

#(vi)_____

# Filter for rows where 'Survived' column is 1 (i.e., passengers who survived)
survived_df = titanic_df[titanic_df['Survived'] == 1]

# Find the oldest, youngest, and average age of the people who survived
oldest_age = survived_df['Age'].max()
youngest_age = survived_df['Age'].min()
average_age = survived_df['Age'].mean()

# Print the results
print("Oldest age of person who survived: {:.2f} years".format(oldest_age))
print("Youngest age of person who survived: {:.2f} years".format(youngest_age))
print("Average age of people who survived: {:.2f} years".format(average_age))
print()

```

Oldest age of person who survived: 80.00 years
 Youngest age of person who survived: 0.42 years
 Average age of people who survived: 28.34 years

vii) The first line of code calls the `pd.crosstab()` function, passing in the 'Sex' column as the index, and the 'Survived' and 'Pclass' columns as the two columns in a list as the columns parameter. The `rownames` parameter is set to 'Sex', and the `colnames` parameter is set to a list of 'Survived' and 'Pclass' as column names.

The resulting cross-tabulation is stored in a new dataframe variable called 'crosstab_df'.

The last line of code prints out the cross-tabulation in a human-readable format using the `print()` function.

Overall, this code is a simple way to summarize the relationships between the 'Sex', 'Survived', and 'Pclass' columns in the "titanic_df" dataset using a cross-tabulation.

```

#(vii)_____

# Create a crosstab of 'Sex', 'Survived', and 'Pclass'
crosstab_df = pd.crosstab(index=titanic_df['Sex'], columns=[titanic_df['Survived'], titanic_df['Pclass']],
                           rownames=['Sex'], colnames=['Survived', 'Pclass'])

# Display the crosstab
print(crosstab_df)
print()

```

Survived	0				1			
Pclass	1	2	3	1	2	3		
Sex								
female	3	6	72	91	70	72		
male	77	91	300	45	17	47		

viii) First, a cross tabulation is created using the `pd.crosstab()` function to count the number of passengers who survived or died based on the 'SibSp' column. The resulting crosstab is stored in a new dataframe variable called 'sibsp_crosstab'.

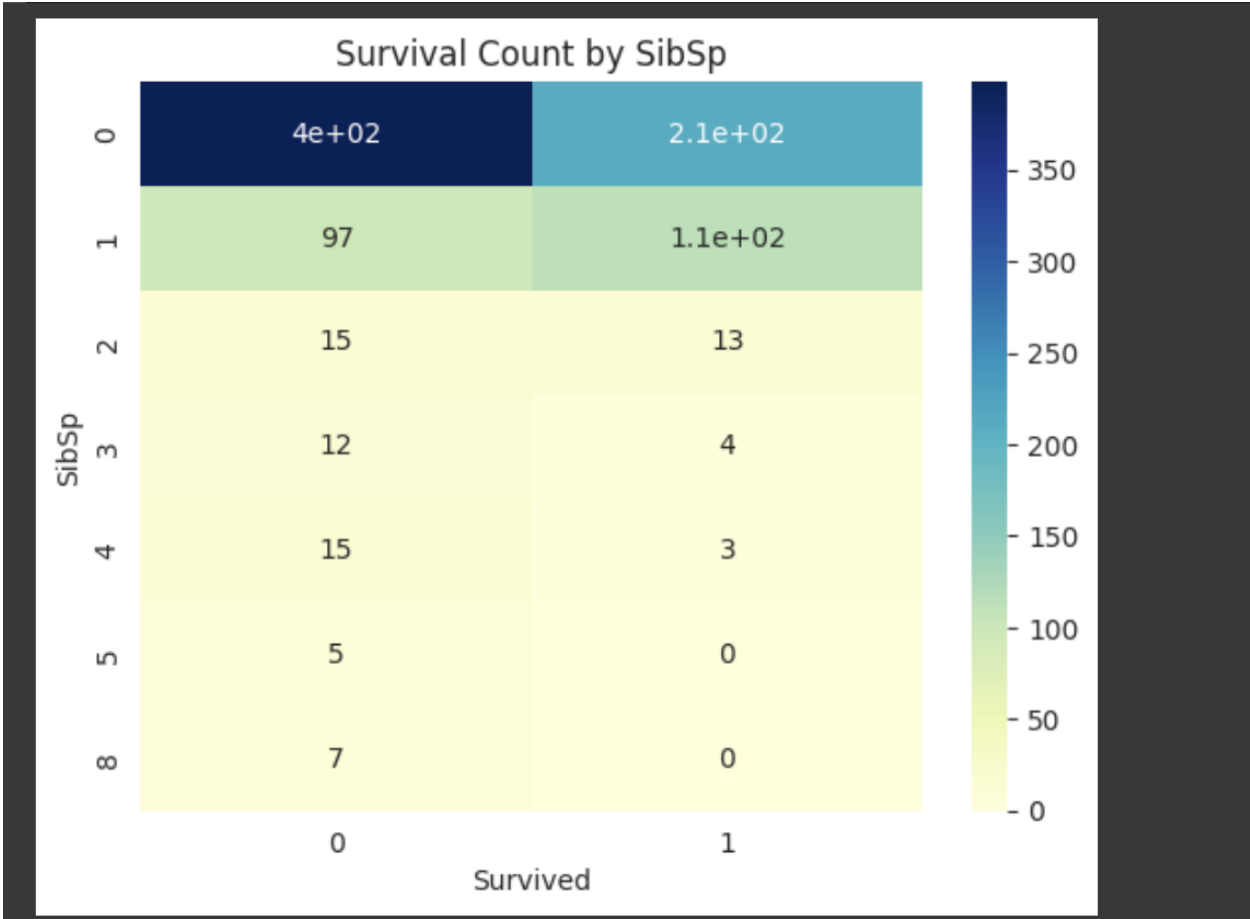
Next, a heatmap is plotted using the seaborn library to visualize the relationship between 'SibSp' and 'Survived'. The heatmap shows the count of survivors and non-survivors for each value of 'SibSp'. Annotations are added to the heatmap to display the count values for each cell. Then, a catplot is plotted using seaborn to visualize the relationship between 'SibSp' and 'Survived' in terms of survival rate. The catplot shows the survival rate for each value of 'SibSp', with error bars indicating the confidence intervals. The resulting plots provide insight into how the number of siblings or spouses on board (SibSp) affected a passenger's chances of survival on the Titanic. Overall, this code provides a good example of how to use data visualization techniques to explore relationships between variables in a dataset.

```
#(viii)_____

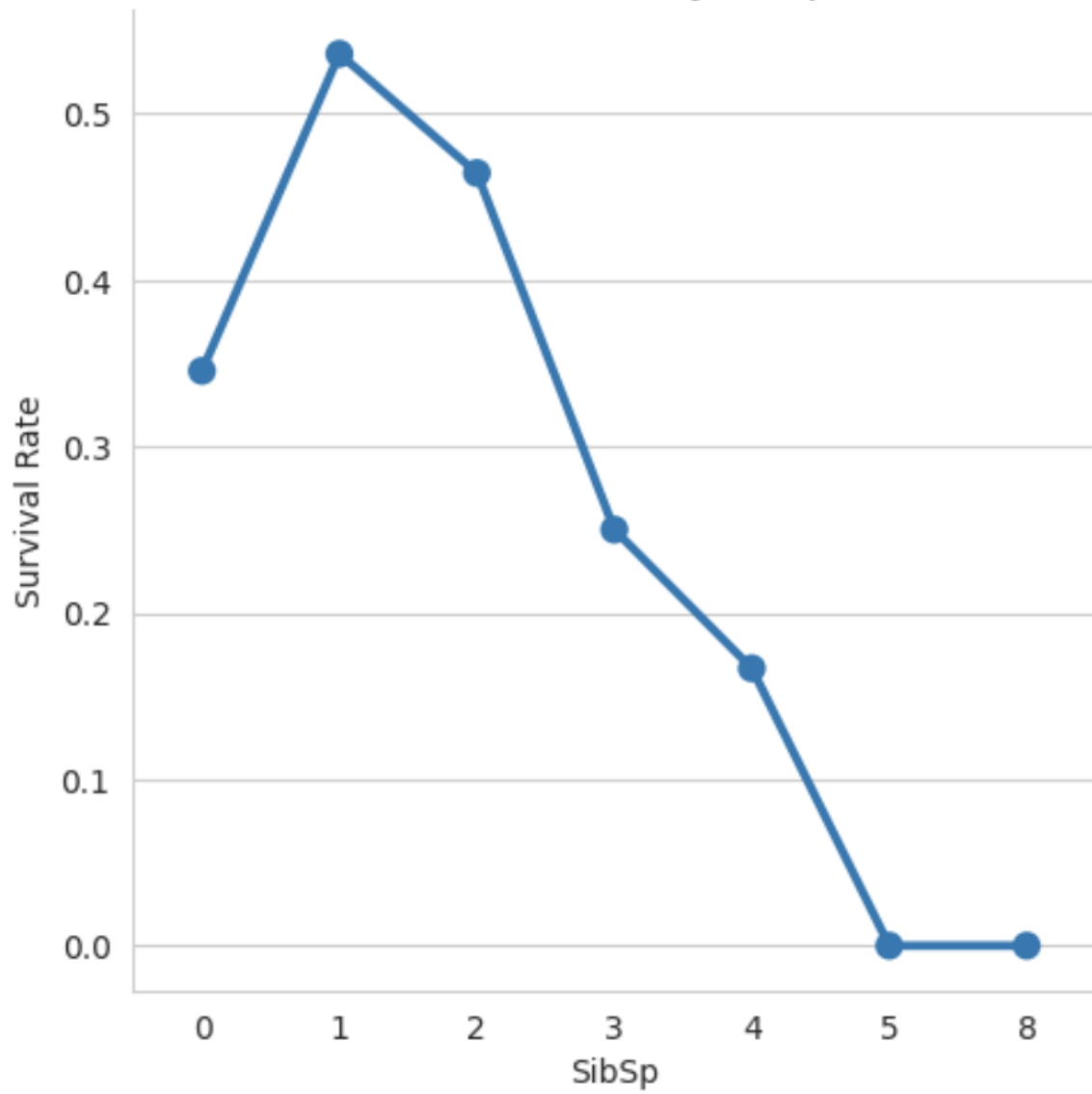
sibsp_crosstab = pd.crosstab(index=titanic_df['SibSp'], columns=titanic_df['Survived'])

# Plot the crosstab
sns.heatmap(sibsp_crosstab, annot=True, cmap='YlGnBu')
plt.title('Survival Count by SibSp')
plt.xlabel('Survived')
plt.ylabel('SibSp')
plt.show()

# Plot catplot of 'SibSp' vs 'Survived'
sns.catplot(x='SibSp', y='Survived', data=titanic_df, kind='point', ci=None)
plt.title('Survival Rate by SibSp')
plt.xlabel('SibSp')
plt.ylabel('Survival Rate')
plt.show()
```



Survival Rate by SibSp



Q2)Prolog music playlist

```
:-dynamic(playlists/2).
^playlist(Name, Songs) :-
    playlists(Name, _),
    write("Playlist already exists"), fail;
    asserta(playlists(Name, Songs)).

add_song(Name, Title, Artist, UpdatedPlaylist) :-
    playlists(Name, Songs),
    (   Songs = []
    -> UpdatedPlaylist = [newsong(Artist, Title)]
    ;   append(Songs, [newsong(Artist, Title)], UpdatedPlaylist)
    ),
    retract(playlists(Name, _)),
    asserta(playlists(Name, UpdatedPlaylist)).

display_playlist(Name) :-
    playlists(Name, X),
    write(Name), nl, displaying(X).

displaying([]) :-!.
displaying([newsong(Artists, Title) | T]) :-
    write(Artists), nl, write(Title), nl,
    displaying(T).
```

This Prolog code describes a playlist management system. Users can make new playlists, add music to pre-existing playlists, and see a playlist's contents. It employs dynamic predicates to store playlists in memory.

The list/2 predicate is declared as dynamic by the dynamic(list/2) statement, which allows its definition to be changed at runtime using the assert and retract statements.

A new playlist with the specified Name and a list of Songs is created using the playlist(Name, Songs) predicate. Using the list/2 predicate, it first determines if a playlist with the specified name already exists. If it does, the condition is false, and a message is printed. If not, asserta/1 is used to insert the new playlist into the database.

A new song with the given Title and Artist is added to an existing playlist with the given Name using the add_song(Name, Title, Artist,

UpdatedPlaylist) predicate. Using the list/2 predicate, it first retrieves the current list of songs for the playlist. It makes a new playlist with the new song if the existing one is empty. If not, it adds the new song to the list already in place. The old playlist is then deleted from the database using retract/1, and the revised playlist is added using asserta/1. A list representing the modified playlist is returned.

To display the contents of a playlist with the specified Name, use the display_playlist(Name) predicate. It calls the displaying/1 predicate to recursively display each song in the list after retrieving the playlist's song list using the list/2 predicate. Recursion is used by the displaying/1 predicate to display the Title and Artist of each song.

```
?- playlist('MyAlbum', []).  
true.  
  
?- add_song('MyAlbum', 'Song1', 'Artist1', UpdatedPlaylist).  
UpdatedPlaylist = [newsong('Artist1', 'Song1')].  
  
?- add_song('MyAlbum', 'Song2', 'Artist2', UpdatedPlaylist).  
UpdatedPlaylist = [newsong('Artist1', 'Song1'), newsong('Artist2', 'Song2')].  
  
?- display_playlist('MyAlbum').  
MyAlbum  
Artist1  
Song1  
Artist2  
Song2  
true.
```

THANK YOU!