YHILLS INTERNSHIP REPORT

SUBMITTED BY
PAILA TEJESWARA RAO (A21126551040)

In fulfillment
of the internship
in

COMPUTER SCIENCE & ENGINEERING(AI&ML,DS)

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Affiliated to Andhra University)
SANGIVALASA, VISAKHAPATNAM – 531162
2021-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(AI,ML&DS)
ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Affiliated to Andhra University)
SANGIVALASA, VISAKHAPATNAM – 531162
2021-2025

BONAFIDE CERTIFICATE

This is to certify that this Internship Report on "Data Science YHILLS" is a bonafide work of P TEJESWARA RAO (A21126551040) of 3/4 CSD carried out the Internship under my Supervision.

Reviewer

Dr. R NDSS KIRAN
Assistant Professor
Department of CSE(AI&ML,DS)

Class Teacher

Mrs. G SURYAKALA ESWARI
Assistant Professor
Department of CSE(AI&ML,DS)

Prof. K. Selvani Deepthi
Head of The Department
Department of CSE(AI&ML,DS)
ANITS

ACKNOWLEDGEMENT

The success of our long-term endeavor is attributed to the advice and support of numerous well-wishers. At this moment, we would like to convey our deep gratitude and appreciation to all those who contributed. Our sincere acknowledgments go to Prof. K. S. Deepthi, who served as the Head of the Department for Computer Science & Engineering (AI&ML,DS) at ANITS, for the invaluable support and guidance provided during our internship.

I wish to express MY sincere thanks and gratitude to our YHILLS Academy for smooth onboarding and analyzing the work associated with internship and for guiding me throughout the internenship. I express my warm and sincere thanks for the encouragement, untiring guidance, and confidence they had shown in me.

I also thank all the staff members of the Computer Science & Engineering (AI&ML,DS) department for their valuable advices. We also thank supporting staff for providing resources as and when required.

P TEJESWARA RAO
A21126551040

Table of Contents:

ABOUT THE COURSE DATA SCIENCE:

TITLE: DATA SCIENCE

WHAT YOU WILL LEARN:

- You'll receive a complete toolkit through this course, enabling your transformation into a data scientist.
- Enhance your resume with sought-after data science skills, including statistical analysis, Python programming using NumPy, pandas, matplotlib, and Seaborn, advanced statistical examination, Tableau, machine learning with statistical models and scikit-learn, and deep learning with TensorFlow.
- Impress surveys by demonstrating your grasp of the field of data science.
- Learn the art of data preprocessing.
- Gain a profound understanding of the science underpinning Machine Learning (an essential aspect that many other courses omit!).

REQUIREMENTS:

- No related knowledge is required. We will begin from the very essentials.
- You'll have to introduce Anaconda. We will tell you the best way to do that bit by bit.

Description:

One of the most promising professions to thrive in this century is that of a data scientist. It is characterized by its focus on programming, innovation, and intelligence. Hence, it comes as no surprise that there has been a surge in demand for data scientists in the job market.

Who is this course for?:

- If you're interested in becoming a Data Scientist or wish to explore the field, consider taking this course.
- If you're seeking a promising career, this course is tailored for you.
- This course is well-suited for beginners since it commences with the fundamentals and steadily enhances your skills.

Data Science Course Syllabus:

MODULE 1: BASICS OF PYTHON

→datatypes
→variables
→data structures (list, tuple, set, dictionary)

MODULE 2: DATA ANALYSIS

→numpy
→pandas

MODULE 3: EXPLORATORY DATA VISULIZATION IN PYTHON

→ Matplotlib
→seaborn

MODULE 4: STATISTICAL THINKING IN PYTHON

→Mesure of central tendency
→Mesure of Dispersion
→ Interquartile Range (IQR)

MODULE 5: SUPERVISED AND UNSUPERVISED LEARNING

→Supervised learning
→unsupervised learning

MODULE 6:MACHINE LEARNING

→ Linear Regression
→Logestic regression
→SVM
→Random forest
→DescisionTree
→K-NN
→adabooster

ASSESMENTS:

After completion of course we submitted 2 projects.

CERTIFICATION:

1. By recognizing your commitment and effort in the program, we are pleased to present you with a course completion certificate from YHills.
2. Upon successfully completing the internship, we offer certificates from both YHills and prestigious universities.
3. As a reward, we furnish you with an industry certificate from well-established and esteemed companies.

Course:

BASIC CONCEPTS IN PYTHON:

- In Python, code blocks are defined through indentation, typically using four spaces. The start of an indented code block is indicated by colons (:), commonly seen in loops and functions.

- Data storage is facilitated by variables, which you can create by assigning a value to a name, as demonstrated with x = 5. Python's dynamic typing eliminates the need to explicitly specify a variable's type.

- Python incorporates various built-in data types, including integers (e.g., 5) and floats (e.g., 3.14). Additionally, it supports strings (e.g., "Hello"), lists (e.g., [1, 2, 3]), tuples (e.g., (1, 2, 3)), and dictionaries (e.g., {'name': 'Alice', 'age': 25}).

- To display output in Python, you can use the print() function. For instance, you can execute print("Hello, World!").

- Python's comment system involves the use of the # symbol. These comments are not processed by the Python interpreter and are primarily utilized for code documentation.

- For decision-making within your code, employ if, elif, and else. For example:

- python
- Copy code
- if x > 5:
- print("x is greater than 5")
- elif x == 5:
- print("x is equal to 5")
- else:
- print("x is less than 5")
-
- Sequences like lists can be iterated over using for loops, while while loops allow for ongoing execution as long as a condition remains true.
-
- Functions in Python enable the encapsulation of code blocks for reusability. You define a function using the def keyword. Here's an example:
-
- python
- Copy code
- def greet(name):
- print(f"Hello, {name}!")
- Lists serve as ordered collections of items, with elements accessible by their index, starting from 0.
-
- Dictionaries, which store key-value pairs, allow access to values using their corresponding keys. They are commonly used for managing related data.
-
- Python's extensive standard library includes various modules that can be imported to access pre-built functions and features. For instance, you can use import math to access mathematical functions.

- File handling in Python entails reading from and writing to files using the open() function. Remember to close the file when you have finished working with it.

-

- Exception handling is crucial for managing errors gracefully and preventing program crashes due to exceptions. You can achieve this using try, except, and finally blocks.

-

- Python supports object-oriented programming (OOP) concepts, such as classes and objects, which assist in organizing and structuring code in a more modular manner.

DATA ANALYSIS(MANIPULATION WITH NUMPY AND PANDAS):

NUMPY:

NumPy is a fundamental package for scientific computing in Python. It provides support for arrays, matrices, and a wide range of mathematical functions to work with them.

1. In Python, code blocks are defined through indentation, typically using four spaces. The start of an indented code block is indicated by colons (:), commonly seen in loops and functions.

3.Data storage is facilitated by variables, which you can create by assigning a value to a name, as demonstrated with x = 5. Python's dynamic typing eliminates the need to explicitly specify a variable's type.

4.Python incorporates various built-in data types, including integers (e.g., 5) and floats (e.g., 3.14). Additionally, it supports strings (e.g., "Hello"), lists (e.g., [1, 2, 3]), tuples (e.g., (1, 2, 3)), and dictionaries (e.g., {'name': 'Alice', 'age': 25}).

5.To display output in Python, you can use the print() function. For instance, you can execute print("Hello, World!").

6.Python's comment system involves the use of the # symbol. These comments are not processed by the Python interpreter and are primarily utilized for code documentation.

7.For decision-making within your code, employ if, elif, and else. For example:

- Lists serve as ordered collections of items, with elements accessible by their index, starting from 0.
-
- Dictionaries, which store key-value pairs, allow access to values using their corresponding keys. They are commonly used for managing related data.
-
- Python's extensive standard library includes various modules that can be imported to access pre-built functions and features. For instance, you can use import math to access mathematical functions.
-
- File handling in Python entails reading from and writing to files using the open() function. Remember to close the file when you have finished working with it.
-
- Exception handling is crucial for managing errors gracefully and preventing program crashes due to exceptions. You can achieve this using try, except, and finally blocks.

-
- Python supports object-oriented programming (OOP) concepts, such as classes and objects, which assist in organizing and structuring code in a more modular manner.

8.NumPy offers a variety of functions for array manipulation, including reshaping, stacking, splitting, and transposing, enabling users to work with arrays in different ways.

9.When it comes to Linear Algebra, NumPy provides built-in support for a range of operations such as matrix multiplication, determinant calculation, eigenvalue computation, and solving linear equations.

10.NumPy's random number generation module (numpy.random) allows users to generate random data or sample from various distributions, adding versatility to data generation tasks.

11.In Python, NumPy seamlessly integrates with other scientific and data analysis libraries like SciPy, Matplotlib, and Pandas, which makes it a critical component in the field of data science and scientific computing.

12.NumPy excels in memory management by efficiently handling memory and supporting data types that are optimized for numerical operations, thus reducing memory overhead.

13.The flexibility in data types provided by NumPy is essential for handling a wide range of data, including integers, floating-point numbers, complex numbers, and custom data types.

14.Being an open-source project, NumPy is both freely available and actively maintained by a dedicated community of developers, ensuring accessibility and continuous improvement

PANDAS:

Pandas, a widely used open-source library for Python, is known for its strength in data manipulation and analysis. It offers user-friendly data structures and functions tailored for structured data, primarily through its two core data structures: DataFrames and Series.

Data Structures:

◦ A DataFrame is a structured data format with labeled rows and columns, resembling a spreadsheet or SQL table. Each column can accommodate varying data types.
◦ A Series, on the other hand, is akin to a DataFrame's column, possessing an associated label or index and structured as a one-dimensional array-like data format.

Data Import/Export:
- Pandas can both import data from a variety of sources, such as CSV, Excel, SQL databases, and more, and also export data to these formats.

Data Cleaning and Transformation:
- Pandas offers robust capabilities for data cleaning and preprocessing, which encompass tasks such as managing missing values, data filtering, and reshaping. Additionally, it enables data transformation and manipulation through functions like merging, joining, and pivoting.

Indexing and Selection:
- You can select and filter data based on labels or conditions using methods like `.loc` and `.iloc`.

- Pandas enables the grouping of data based on specific criteria and executing aggregation functions on the grouped data.

- In the realm of time series data, Pandas boasts robust support. It provides functions for resampling, time-based indexing, and calculating moving window statistics.

- While Pandas is not a dedicated visualization library, it can effortlessly collaborate with visualization libraries like Matplotlib and Seaborn for data plotting.

- As it's built on top of NumPy, Pandas can seamlessly integrate with NumPy arrays to streamline data processing.

- Pandas offers performance optimization functions, including vectorized operations, which deliver significantly faster results compared to traditional for-loops.

- For identifying and managing missing or NaN values in data, Pandas provides a range of methods.

- It can handle reading and writing data in various file formats, simplifying the process of working with data from diverse sources.

- Thanks to its flexibility and user-friendliness, Pandas has gained wide adoption in the fields of data analysis, data science, and machine learning.

- Pandas boasts a thriving and extensive community, and numerous other data-related libraries and tools have been designed to seamlessly integrate with it.

EXPLORATORY DATA VISULIZATION IN PYTHON WITH MATPLOTLIB AND SEABORN:

Matplotlib:

Matplotlib, a versatile and powerful Python library for data visualization, offers a low-level interface to create a wide range of plots and charts. Let's delve deeper into Matplotlib for a more comprehensive understanding:

Line Plot:
- A line plot is created using `plt.plot()`.
- It's useful for visualizing trends or changes in data over time or continuous variables.
- You can customize the line style, color, and markers.

Scatter Plot:
- Scatter plots are used to display individual data points on a two-dimensional space.
- They are great for showing the relationships and distributions of data points.
- You can use `plt.scatter()` to create scatter plots.

Bar Plot:

- Bar plots are useful for showing comparisons between different categories or groups.
- They can be created using `plt.bar()` or `plt.barh()` for horizontal bar plots.

Histogram:

- Histograms show the distribution of a single variable.
- They are created with `plt.hist()` and are useful for visualizing data frequency.

Box Plot:

- Box plots (box-and-whisker plots) display the distribution of a dataset's summary statistics.
- They are created using `plt.boxplot()` and are helpful for identifying outliers and assessing data distributions.

Seaborn:

Seaborn is built on top of Matplotlib and provides a high-level interface for creating aesthetically pleasing statistical visualizations. Here's a more detailed explanation:

Heatmap:

- Heatmaps are used to visualize the relationships between two variables in a matrix format.
- Seaborn's `sns.heatmap()` makes it easy to create heatmaps with color-coded values.

Pair Plot:

- Pair plots, created using `sns.pairplot()`, are a grid of scatterplots for visualizing relationships between multiple variables in a dataset.
- They are especially useful for understanding pairwise correlations.

Violin Plot:

- Violin plots are a combination of box plots and kernel density estimation plots.
- They show the distribution of data across different categories or groups.
- `sns.violinplot()` is used to create them.

Count Plot:

- Count plots are a type of bar plot that shows the count of observations in each category.
- They can be generated with `sns.countplot()` and are great for categorical data.

Facet Grid:

- Seaborn's `FacetGrid` allows you to create a grid of plots based on the structure of your dataset.
- You can use it to visualize relationships within subgroups of your data.

In summary, Matplotlib is a low-level library that gives you full control over the customization of your plots, while Seaborn is a high-level library designed for quick and informative statistical visualization. The choice between them depends on your specific visualization needs, level of customization, and the type of plots you want to create.

STATISTICAL THINKING IN PYTHON:

Python's statistical thinking encompasses a range of concepts and techniques used to analyze and summarize data. The three fundamental concepts include the measures of central tendency, measures of dispersion, and the inter quartile range (IQR).

Measures of Central Tendency:

1. **Mean (Average):**

- The most commonly used measure of central tendency is the mean, which is obtained by summing up all the data points and then dividing the sum by the total number of data points. To calculate the mean in Python, you can utilize the numpy library with the following code:
-

```
import numpy as np
mean = np.mean(data)
```

2.Median:

- The median, when the data is sorted in ascending order, represents the middle value of a dataset. When there's an even number of data points, it is the average of the two middle values.
- To compute the median with `numpy`:

```
python
median = np.median(data)
```

3.Mode:

- The mode is the value that appears most frequently in the dataset, and a dataset can be either uni-modal with only one mode or multi modal with more than one mode, or it may have no mode at all.

- You can calculate the mode using libraries like `statistics` or `scipy.stats`:

```
python
from statistics import mode
from scipy import stats
mode_value = mode(data)
mode_value = stats.mode(data)
```

Measures of Dispersion:

1.Range:

- The range measures the spread of data by finding the difference between the maximum and minimum values in the dataset.
- To calculate the range:

```
python
```

data_range = max(data) - min(data)

2.Variance and Standard Deviation:

- Variance quantifies how data points deviate from the mean. A higher variance indicates greater spread.
- Standard deviation is the square root of the variance and is a more interpretable measure.
- You can compute variance and standard deviation using `numpy`:

python

```
variance = np.var(data)
std_deviation = np.std(data)
```

Interquartile Range (IQR):

-The IQR measures the spread of data within the middle 50% of the dataset, making it a robust measure that is not affected by extreme outliers.

- To compute the IQR using `numpy`:

python

```
Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)
IQR = Q3 - Q1
```

- The IQR is particularly useful for identifying potential outliers and creating box plots. It defines the box (representing the middle 50% of data, from Q1 to Q3) and the whiskers (commonly 1.5 times the IQR length).
- Values that fall below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR are often considered potential outliers.

In summary, measures of central tendency help you understand the typical or central values in your data, while measures of dispersion provide insights into how the data is spread out. The IQR, as a measure of dispersion, focuses on the middle 50% of the data and is useful for outlier detection. Python libraries like `numpy`, `statistics`, and `scipy` make it easy to calculate these statistical measures for your datasets.

SUPERVISED AND UNSUPERVISED LEARNING:

Supervised:

Supervised learning is a type of machine learning where the algorithm learns from a labeled dataset. In a supervised learning problem, the dataset used for training consists of input-output pairs, where the output (target) is known. The goal is to learn a mapping from inputs to outputs so that the algorithm can make predictions or classifications on new, unseen data.

1. **Training Data**: In supervised learning, you start with a training dataset that includes input features and corresponding target values. For example, if you're building a spam email classifier, the input features could be the email's content, and the target values would be whether the email is spam or not (binary classification).

2. **Model Training**: You choose a machine learning model (e.g., linear regression, decision trees, neural networks) and train it on the training data. The model learns to make predictions based on the patterns it discovers in the input-output pairs.

3. **Prediction**: After the model is trained, you can use it to make predictions or classify new, unseen data. For example, if you have a new email, the model can predict whether it's spam or not based on the learned patterns.

4. **Evaluation**: You assess the model's performance by comparing its predictions to the true target values in a separate evaluation dataset. Common metrics for evaluation include accuracy, precision, recall, and F1-score, depending on the type of problem (classification or regression).

5. **Use Cases**: Supervised learning is used for various tasks, such as image classification, sentiment analysis, recommendation systems, and more. It's well-suited for problems where you have labeled data and want to make predictions or decisions.

Unsupervised:

Unsupervised learning, on the other hand, deals with datasets that have no labeled outputs. The primary goal of unsupervised learning is to find underlying patterns, structures, or relationships in the data without guidance on what the output should be. It's about discovering hidden insights or grouping similar data points together.

1. **Data Clustering**: A common task in unsupervised learning is data clustering. Clustering algorithms group similar data points together based on their similarities or distances, creating clusters or groups. The most famous clustering algorithm is K-Means.

2. **Dimensionality Reduction**: Unsupervised learning also includes dimensionality reduction techniques, such as Principal Component Analysis (PCA) and t-SNE. These methods help reduce the complexity of data by capturing its essential information in a lower-dimensional space.

3. **Anomaly Detection**: Another application is anomaly detection, where the algorithm identifies unusual data points or outliers in the dataset. This is helpful for fraud detection or quality control.

4. **Generative Modeling**: Unsupervised learning can be used to create generative models, such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), which can generate new data points that resemble the training data.

5. **Use Cases**: Unsupervised learning is employed in various domains, including customer segmentation, data compression, and exploratory data analysis. It's particularly useful when you have large, unlabeled datasets and want to discover patterns or reduce data complexity.

supervised learning involves learning from labeled data to make predictions or classifications, while unsupervised learning is about discovering hidden patterns or structures in unlabeled data. Both paradigms play crucial roles in machine learning and have diverse applications across various domains.

MACHINE LEARNING MODELS:

1. **Linear Regression**:
   - Linear regression is used for regression tasks, where the goal is to predict a continuous numerical output based on input features.
   - The model assumes a linear relationship between the input features and the target variable. It finds the best-fit line that minimizes the sum of squared differences between predicted and actual values.

2. **Logistic Regression**:
   - Logistic regression is primarily used for binary classification problems, where the output is a binary (two-class) label (e.g., 0 or 1).
   - It uses the logistic function to model the probability of the input belonging to one of the classes.

3. **Decision Trees**:
   - Decision trees are versatile for both classification and regression. They create a tree-like structure where each node represents a decision based on input features.

4.Random Forest:

- Random Forest is an ensemble method that combines multiple decision trees to improve predictive accuracy and reduce overfitting.
- It builds multiple decision trees and aggregates their predictions, often using a majority vote (for classification) or averaging (for regression).

4. **Support Vector Machines (SVM):**
- SVM is used for both classification and regression. It aims to find a hyperplane that best separates classes or predicts continuous values.
- SVM seeks the optimal hyperplane by maximizing the margin between data points from different classes. It can use various kernel functions to handle non-linear data.

6. **K-Nearest Neighbors (K-NN):**
- K-NN is a simple yet effective algorithm for both classification and regression. It makes predictions based on the majority class or average of the k-nearest data points in the feature space.
- The choice of 'k' (number of neighbors) impacts the algorithm's performance and sensitivity to noise.

7. **Naive Bayes**:
- Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It is often used in text classification and spam filtering.
- It assumes that features are conditionally independent given the class, simplifying the computation of probabilities.

8. **AdaBoost**:
- AdaBoost is an ensemble learning method that combines multiple weak classifiers to create a strong classifier.
- It assigns weights to data points and repeatedly trains models to correct the errors made by the previous models. The final model is a weighted sum of the weak models.

PROJECT:

TITLE : TAXI_FARE

IMPORTING NECESSARY LIBRARIES

In [1]:
```
import pandas as pd
pd.set_option("display.max_columns",None)

import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import math

import seaborn as sns

from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn import svm

from sklearn.model_selection import train_test_split
```

RETRIVING AND EXPLORING DATASET

In [2]:
```
df=pd.read_csv('/home/bot/Desktop/intership s/yhills/projects/taxifare/taxifare.csv')
```

In [3]:
```
df
```

Out[3]:

	unique_id	amount	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger
0	26:21.0	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1
1	52:16.0	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1
2	35:00.0	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2
3	30:42.0	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	1
4	51:00.0	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1
...	...	...	...	...	...	...	...	...
49995	25:15.0	15.0	2013-06-12 23:25:15 UTC	-73.999973	40.748531	-74.016899	40.705993	1
49996	19:18.0	7.5	2015-06-22 17:19:18 UTC	-73.984756	40.768211	-73.987366	40.760597	1
49997	53:00.0	6.9	2011-01-30 04:53:00 UTC	-74.002698	40.739428	-73.998108	40.759483	1
49998	09:00.0	4.5	2012-11-06 07:09:00 UTC	-73.946062	40.777567	-73.953450	40.779687	2
49999	13:14.0	10.9	2010-01-13 08:13:14 UTC	-73.932603	40.763805	-73.932603	40.763805	1

In [4]:
```
df.drop(['unique_id'],axis=1,inplace=True)
```

In [5]:
```
df.head(10)
```

Out[5]:

	amount	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger
0	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1
1	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1
2	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2
3	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	1
4	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1
5	12.1	2011-01-06 09:50:45 UTC	-74.000964	40.731630	-73.972892	40.758233	1
6	7.5	2012-11-20 20:35:00 UTC	-73.980002	40.751662	-73.973802	40.764842	1
7	16.5	2012-01-04 17:22:00 UTC	-73.951300	40.774138	-73.990095	40.751048	1
8	9.0	2012-12-03 13:10:00 UTC	-74.006462	40.726713	-73.993078	40.731628	1
9	8.9	2009-09-02 01:11:00 UTC	-73.980658	40.733873	-73.991540	40.758138	2

In [6]:
```
df.columns
```

Out[6]:
```
Index(['amount', 'date_time_of_pickup', 'longitude_of_pickup',
       'latitude_of_pickup', 'longitude_of_dropoff', 'latitude_of_dropoff',
       'no_of_passenger'],
      dtype='object')
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   amount               50000 non-null  float64
 1   date_time_of_pickup  50000 non-null  object 
 2   longitude_of_pickup  50000 non-null  float64
 3   latitude_of_pickup   50000 non-null  float64
 4   longitude_of_dropoff 50000 non-null  float64
 5   latitude_of_dropoff  50000 non-null  float64
 6   no_of_passenger      50000 non-null  int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 2.7+ MB
```

```
In [8]: df.isna().sum()

Out[8]: amount                0
        date_time_of_pickup   0
        longitude_of_pickup   0
        latitude_of_pickup    0
        longitude_of_dropoff  0
        latitude_of_dropoff   0
        no_of_passenger       0
        dtype: int64
```

CHANGING FORMATS FOR CALCULATIONS

```
In [9]: df['date_time_of_pickup'] = df['date_time_of_pickup'].str.replace(' UTC', '')
```

```
In [10]: df['date_time_of_pickup']= pd.to_datetime(df['date_time_of_pickup']).dt.round('H')
         df
```

Out[10]:

	amount	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger
0	4.5	2009-06-15 17:00:00	-73.844311	40.721319	-73.841610	40.712278	1
1	16.9	2010-01-05 17:00:00	-74.016048	40.711303	-73.979268	40.782004	1
2	5.7	2011-08-18 01:00:00	-73.982738	40.761270	-73.991242	40.750562	2
3	7.7	2012-04-21 05:00:00	-73.987130	40.733143	-73.991567	40.758092	1
4	5.3	2010-03-09 08:00:00	-73.968095	40.768008	-73.956655	40.783762	1
...	...	...	...	...	...	...	...
49995	15.0	2013-06-12 23:00:00	-73.999973	40.748531	-74.016899	40.705993	1
49996	7.5	2015-06-22 17:00:00	-73.984756	40.768211	-73.987366	40.760597	1
49997	6.9	2011-01-30 05:00:00	-74.002698	40.739428	-73.998108	40.759483	1
49998	4.5	2012-11-06 07:00:00	-73.946062	40.777567	-73.953450	40.779687	2
49999	10.9	2010-01-13 08:00:00	-73.932603	40.763805	-73.932603	40.763805	1

50000 rows × 7 columns

```
In [11]: df['date_time_of_pickup'] = df['date_time_of_pickup'].astype(str).str[11:13]
         df
```

Out[11]:

	amount	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger
0	4.5	17	-73.844311	40.721319	-73.841610	40.712278	1
1	16.9	17	-74.016048	40.711303	-73.979268	40.782004	1
2	5.7	01	-73.982738	40.761270	-73.991242	40.750562	2
3	7.7	05	-73.987130	40.733143	-73.991567	40.758092	1
4	5.3	08	-73.968095	40.768008	-73.956655	40.783762	1
...	...	...	...	...	...	...	...
49995	15.0	23	-73.999973	40.748531	-74.016899	40.705993	1
49996	7.5	17	-73.984756	40.768211	-73.987366	40.760597	1
49997	6.9	05	-74.002698	40.739428	-73.998108	40.759483	1
49998	4.5	07	-73.946062	40.777567	-73.953450	40.779687	2
49999	10.9	08	-73.932603	40.763805	-73.932603	40.763805	1

50000 rows × 7 columns

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   amount                50000 non-null  float64
 1   date_time_of_pickup   50000 non-null  object 
 2   longitude_of_pickup   50000 non-null  float64
 3   latitude_of_pickup    50000 non-null  float64
 4   longitude_of_dropoff  50000 non-null  float64
 5   latitude_of_dropoff   50000 non-null  float64
 6   no_of_passenger       50000 non-null  int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 2.7+ MB
```

In [13]:
```
df['date_time_of_pickup']=df['date_time_of_pickup'].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   amount                50000 non-null  float64
 1   date_time_of_pickup   50000 non-null  int64  
 2   longitude_of_pickup   50000 non-null  float64
 3   latitude_of_pickup    50000 non-null  float64
 4   longitude_of_dropoff  50000 non-null  float64
 5   latitude_of_dropoff   50000 non-null  float64
 6   no_of_passenger       50000 non-null  int64  
dtypes: float64(5), int64(2)
memory usage: 2.7 MB
```

In [14]:
```
df
```

Out[14]:

	amount	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger
0	4.5	17	-73.844311	40.721319	-73.841610	40.712278	1
1	16.9	17	-74.016048	40.711303	-73.979268	40.782004	1
2	5.7	1	-73.982738	40.761270	-73.991242	40.750562	2
3	7.7	5	-73.987130	40.733143	-73.991567	40.758092	1
4	5.3	8	-73.968095	40.768008	-73.956655	40.783762	1
...	...	...	...	...	...	...	...
49995	15.0	23	-73.999973	40.748531	-74.016899	40.705993	1
49996	7.5	17	-73.984756	40.768211	-73.987366	40.760597	1
49997	6.9	5	-74.002698	40.739428	-73.998108	40.759483	1
49998	4.5	7	-73.946062	40.777567	-73.953450	40.779687	2
49999	10.9	8	-73.932603	40.763805	-73.932603	40.763805	1

50000 rows × 7 columns

METHODS TO CHANGE TO THOSE GIVEN LATITUDE AND LONGITUDES TO DISTANCE:

```
In [16]: from math import radians, cos, sin, asin, sqrt

         def haversine(k):
             lon1, lat1, lon2, lat2 = map(radians, [k[0], k[1], k[2], k[3]])
             dlon = lon2 - lon1
             dlat = lat2 - lat1
             a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
             c =  2 * asin(sqrt(a))
             return 6371* c
```

```
In [19]: df.drop(['longitude_of_pickup','latitude_of_pickup','longitude_of_dropoff','latitude_of_dropoff'],axis=1,inplace=Tru
```

```
In [18]: df
```

Out[18]:

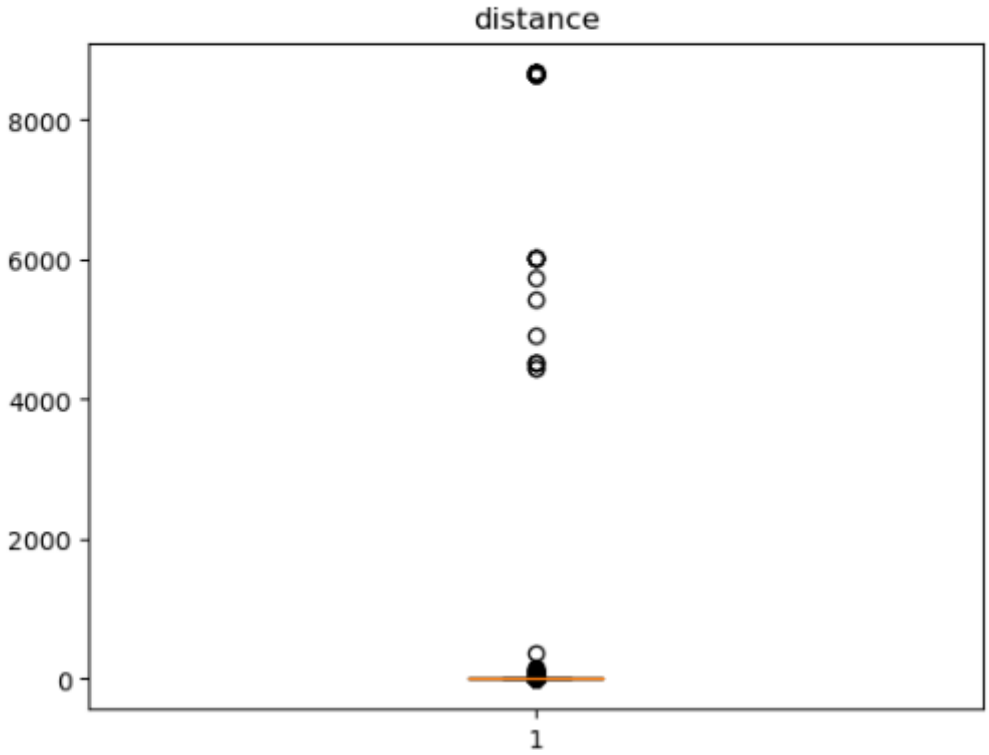
kup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger	distance
311	40.721319	-73.841610	40.712278	1	1.030764
048	40.711303	-73.979268	40.782004	1	8.450134
738	40.761270	-73.991242	40.750562	2	1.389525
130	40.733143	-73.991567	40.758092	1	2.799270
095	40.768008	-73.956655	40.783762	1	1.999157
...	...	...	...	...	...
973	40.748531	-74.016899	40.705993	1	4.940374
756	40.768211	-73.987366	40.760597	1	0.874706
698	40.739428	-73.998108	40.759483	1	2.263286
062	40.777567	-73.953450	40.779687	2	0.665245
603	40.763805	-73.932603	40.763805	1	0.000000

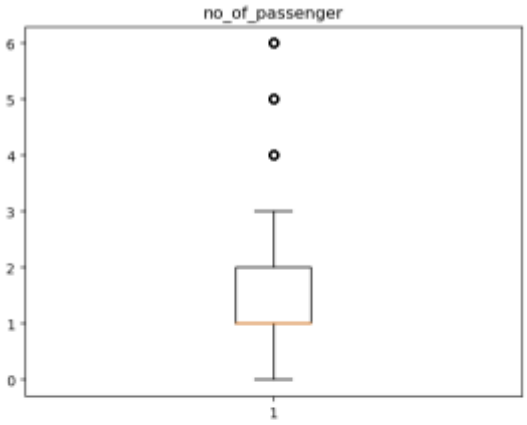
- DROP THE COORDINATES AS WE GOT DISTANCE

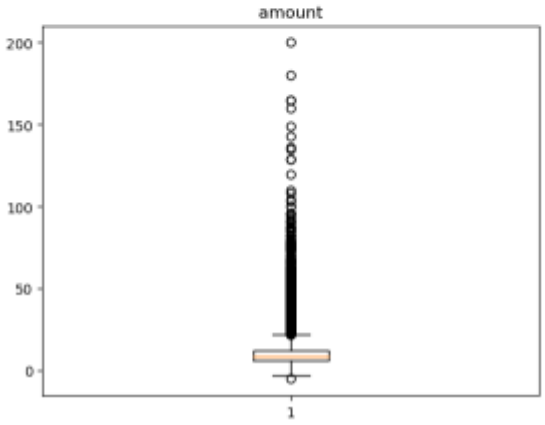
VISUALIZATION OF OUTLIER ANALYSIS

```
[20]: l1=['distance','date_time_of_pickup','no_of_passenger','amount']
```

In [21]:
```
for i in l1:
    plt.boxplot(df[f'{i}'])
    plt.title(i)
    plt.show()
```







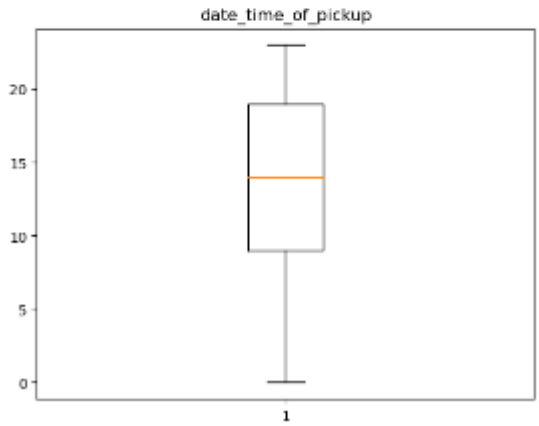
REMOVING OUTLIERS

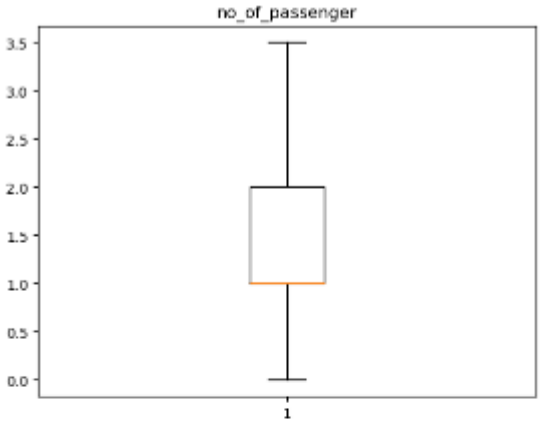
In [22]:
```
def remove_outlier(col):
    sorted(col)
    q1=col.quantile(0.25)
    q3=col.quantile(0.75)
    iqr=q3-q1
    lowerRange=q1-(1.5*iqr)
    higherRange=q3+(1.5*iqr)
    return lowerRange,higherRange
```

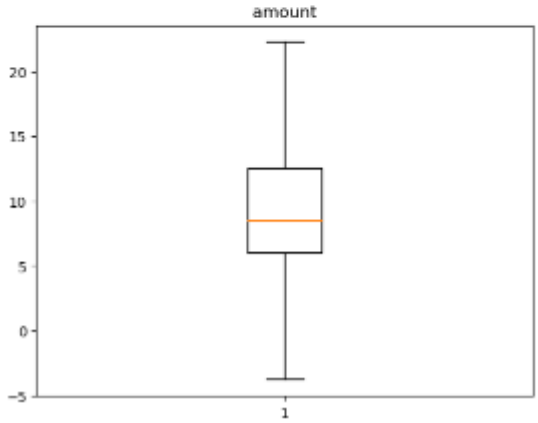
In [23]:
```
l2=['distance','no_of_passenger','amount']
```

In [24]:
```
for i in l2:
    l,u=remove_outlier(df[f'{i}'])
    df[f'{i}']=np.where(df[f'{i}']>u,u,df[f'{i}'])
    df[f'{i}']=np.where(df[f'{i}']<l,l,df[f'{i}'])
```

In [25]:
```
for i in l1:
    plt.boxplot(df[f'{i}'])
    plt.title(i)
    plt.show()
```

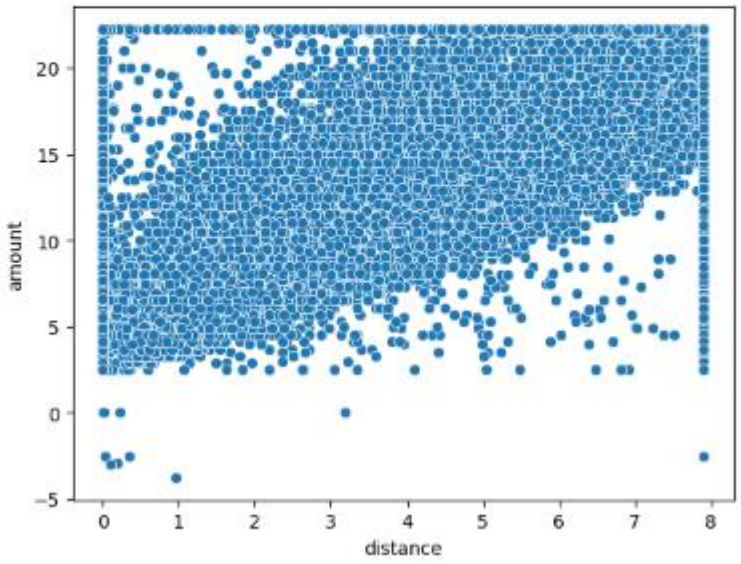


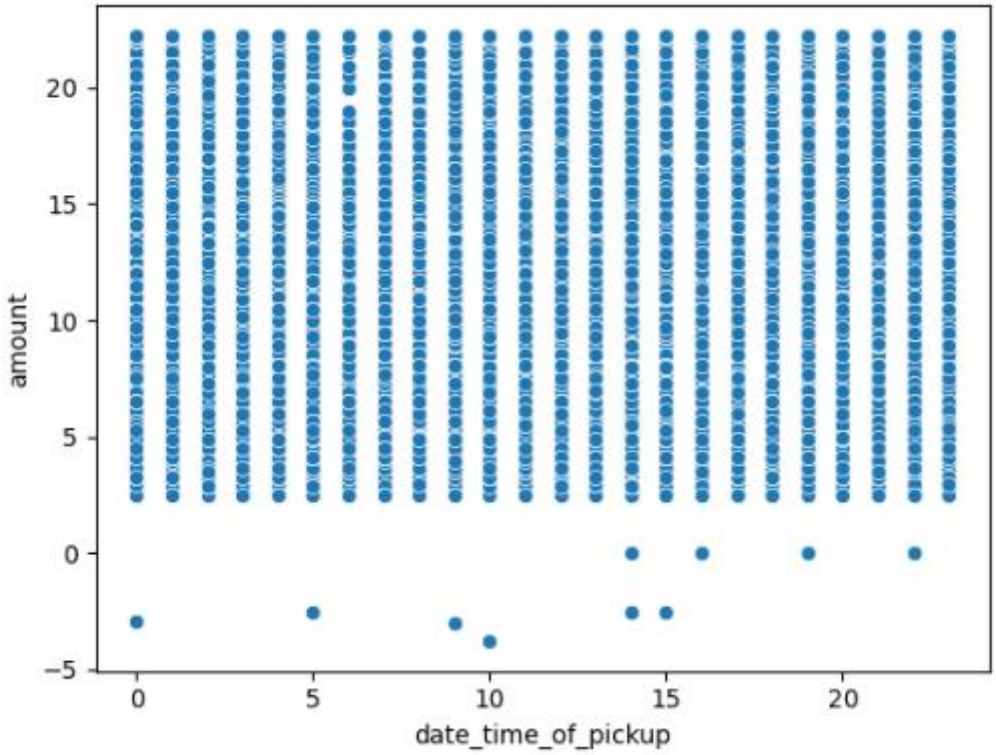


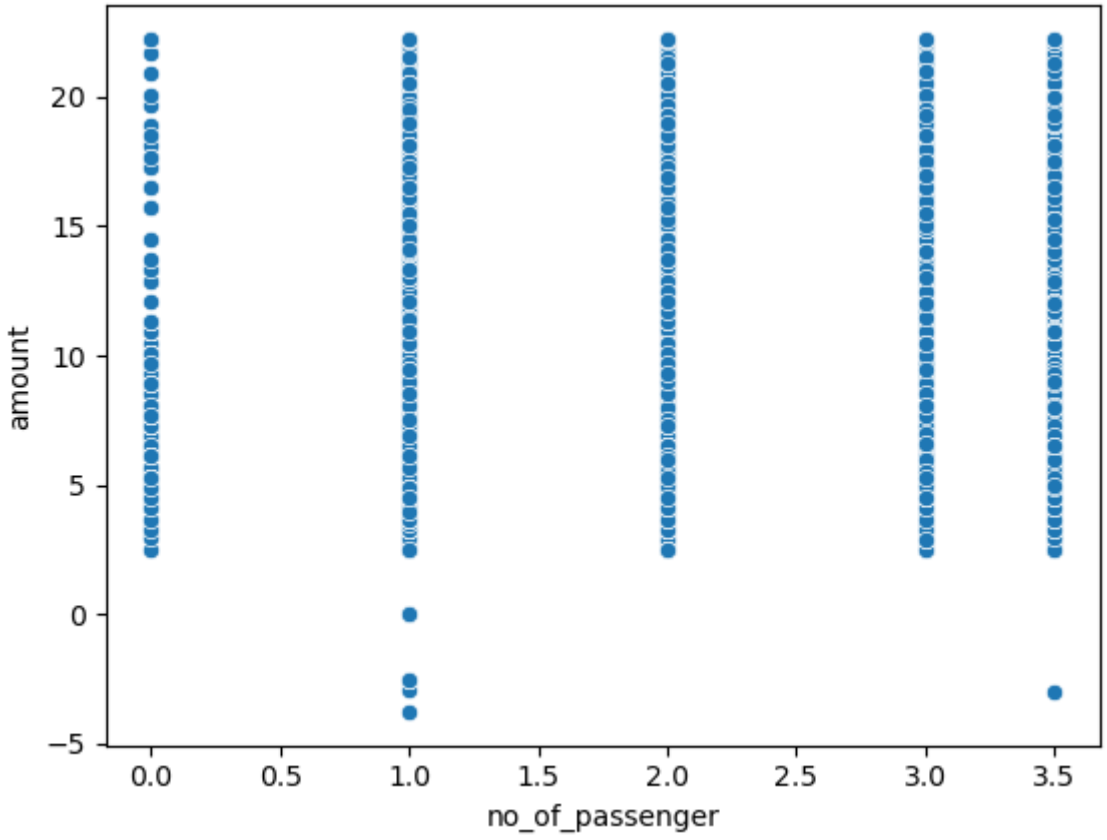


VISUALIZATION OF ANALYSIS

In [26]:
```
ll.pop()
for i in ll:
    sns.scatterplot(x=df[f'{i}'],y=df.amount)
    plt.show()
```




amount
20
15
10
5
0
−5
0
5
10
15
20
date_time_of_pickup


amount
20
15
10
5
0
−5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
no_of_passenger

MODEL TRAINING AND TESTING

In [27]:
```
reg=LinearRegression()
```

In [35]:
```
x,X,y,Y=train_test_split(df.drop(['amount'],axis=1),df['amount'],test_size=0.05)
```

In [36]:
```
reg.fit(x,y)
```

Out[36]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [37]:
```
reg.score(x,y)
```

Out[37]:
```
0.7116921405489824
```

In [38]:
```
reg.score(X,Y)
```

Out[38]:
```
0.7087810496562208
```

GETTING THE AVERAGE OF 5 TESTS FOR ACCURACY

In [39]:
```
test_score=[]
train_score=[]
for _ in range(1):
    x,X,y,Y=train_test_split(df.drop(['amount'],axis=1),df['amount'],test_size=0.05)
    reg.fit(x,y)
    train_score.append(reg.score(x,y))
    test_score.append(reg.score(X,Y))
print("train_socre:",np.mean(train_score),"test_score",np.mean(test_score))
```

```
train_socre: 0.7114726389967938 test_score 0.7130932911646355
```

the difference between train and test score is acceptable and the model predicts with an accuracy of 70+ %

TITLE: H1N1_VACCINATION PREDICTION :

IMPORTING NECESSARY LIBRARIES

In [1]:
```
import pandas as pd
pd.set_option("display.max_columns",None)

import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import math

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

from sklearn.tree import DecisionTreeClassifier

from sklearn import svm
```

RETRIVING AND EXPLORING DATASET

In [2]:
```
df=pd.read_csv('/home/bot/Desktop/intership/yhills/projects/vaccine/vaccine.csv')
```

In [3]:
```
df
```

Out[3]:

	unique_id	h1n1_worry	h1n1_awareness	antiviral_medication	contact_avoidance	bought_face_mask	wash_hands_frequently	avoid_large_gatherings	r
0	0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	1	3.0	2.0	0.0	1.0	0.0	1.0	0.0	
2	2	1.0	1.0	0.0	1.0	0.0	0.0	0.0	
3	3	1.0	1.0	0.0	1.0	0.0	1.0	1.0	
4	4	2.0	1.0	0.0	1.0	0.0	1.0	1.0	
...	...	...	...	...	...	...	...	...	...
26702	26702	2.0	0.0	0.0	1.0	0.0	0.0	0.0	
26703	26703	1.0	2.0	0.0	1.0	0.0	1.0	0.0	

```
In [4]: df.drop(['unique_id'],axis=1,inplace=True)

In [5]: df.shape
Out[5]: (26707, 33)

In [6]: df.columns
Out[6]: Index(['h1n1_worry', 'h1n1_awareness', 'antiviral_medication',
               'contact_avoidance', 'bought_face_mask', 'wash_hands_frequently',
               'avoid_large_gatherings', 'reduced_outside_home_cont',
               'avoid_touch_face', 'dr_recc_h1n1_vacc', 'dr_recc_seasonal_vacc',
               'chronic_medic_condition', 'cont_child_undr_6_mnths',
               'is_health_worker', 'has_health_insur', 'is_h1n1_vacc_effective',
               'is_h1n1_risky', 'sick_from_h1n1_vacc', 'is_seas_vacc_effective',
               'is_seas_risky', 'sick_from_seas_vacc', 'age_bracket', 'qualification',
               'race', 'sex', 'income_level', 'marital_status', 'housing_status',
               'employment', 'census_msa', 'no_of_adults', 'no_of_children',
               'h1n1_vaccine'],
              dtype='object')

In [7]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26707 entries, 0 to 26706
Data columns (total 33 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   h1n1_worry                   26615 non-null  float64
 1   h1n1_awareness               26591 non-null  float64
 2   antiviral_medication         26636 non-null  float64
 3   contact_avoidance            26499 non-null  float64
 4   bought_face_mask             26688 non-null  float64
 5   wash_hands_frequently        26665 non-null  float64
 6   avoid_large_gatherings       26620 non-null  float64
 7   reduced_outside_home_cont    26625 non-null  float64
 8   avoid_touch_face             26579 non-null  float64
 9   dr_recc_h1n1_vacc            24547 non-null  float64
 10  dr_recc_seasonal_vacc        24547 non-null  float64
 11  chronic_medic_condition      25736 non-null  float64
 12  cont_child_undr_6_mnths      25887 non-null  float64
 13  is_health_worker             25903 non-null  float64
 14  has_health_insur             14433 non-null  float64
 15  is_h1n1_vacc_effective       26316 non-null  float64
 16  is_h1n1_risky                26319 non-null  float64
 17  sick_from_h1n1_vacc          26312 non-null  float64
 18  is_seas_vacc_effective       26245 non-null  float64

In [8]: df.isna().sum()
Out[8]: h1n1_worry                     92
        h1n1_awareness                116
        antiviral_medication           71
        contact_avoidance             208
        bought_face_mask               19
        wash_hands_frequently          42
        avoid_large_gatherings         87
        reduced_outside_home_cont      82
        avoid_touch_face              128
        dr_recc_h1n1_vacc            2160
        dr_recc_seasonal_vacc        2160
        chronic_medic_condition       971
        cont_child_undr_6_mnths       820
        is_health_worker              804
        has_health_insur            12274
        is_h1n1_vacc_effective        391
        is_h1n1_risky                 388
        sick_from_h1n1_vacc           395
        is_seas_vacc_effective        462
        is_seas_risky                 514
        sick_from_seas_vacc           537
```

- dropping has_health_insur because it have a high amount of null values.
- By using the below method int and float variable having null values are replaced by median and object type are replaced by mode.

```
In [9]: df.drop(['has_health_insur'],axis=1,inplace=True)
```

```
In [10]: for i in df:
             if df[f'{i}'].dtype==float or df[f'{i}'].dtype==int:
                 df[f'{i}'].replace(np.nan,np.mean(df[f'{i}']),inplace=True)
             else:
                 df[f'{i}'].replace(np.nan,df[f'{i}'].mode()[0],inplace=True)
```

- Null values are removed.

```
In [11]: df.isna().sum()
```

```
Out[11]: h1n1_worry                    0
         h1n1_awareness                0
         antiviral_medication          0
         contact_avoidance             0
         bought_face_mask              0
         wash_hands_frequently         0
         avoid_large_gatherings        0
         reduced_outside_home_cont     0
         avoid_touch_face              0
         dr_recc_h1n1_vacc             0
         dr_recc_seasonal_vacc         0
         chronic_medic_condition       0
         cont_child_undr_6_mnths       0
         is_health_worker              0
         is_h1n1_vacc_effective        0
         is_h1n1_risky                 0
```

- Changing object type variable to integer using label encoder.

```
In [13]: for i in df:
             if df[f'{i}'].dtype==object:
                 df[f'{i}']=le.fit_transform(df[f'{i}'])
```

```
In [14]: df.head(10)
```

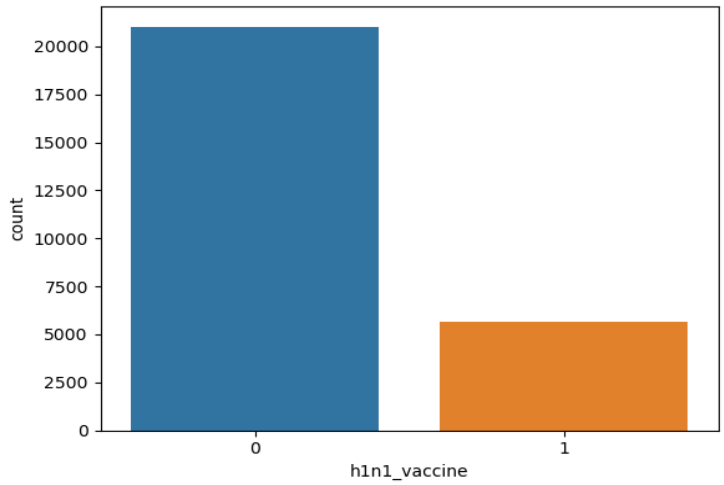
VISUALIZATION

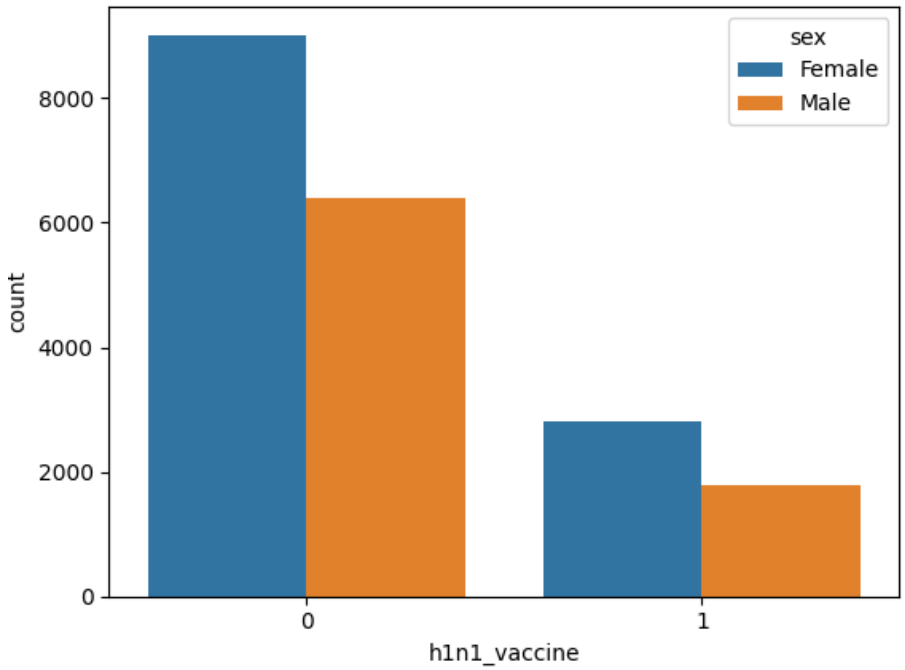
```
In [15]: for i in df.drop(['h1n1_vaccine'],axis=1):
             sns.scatterplot(x=df[f'{i}'],y=df.h1n1_vaccine)
             plt.show()
```

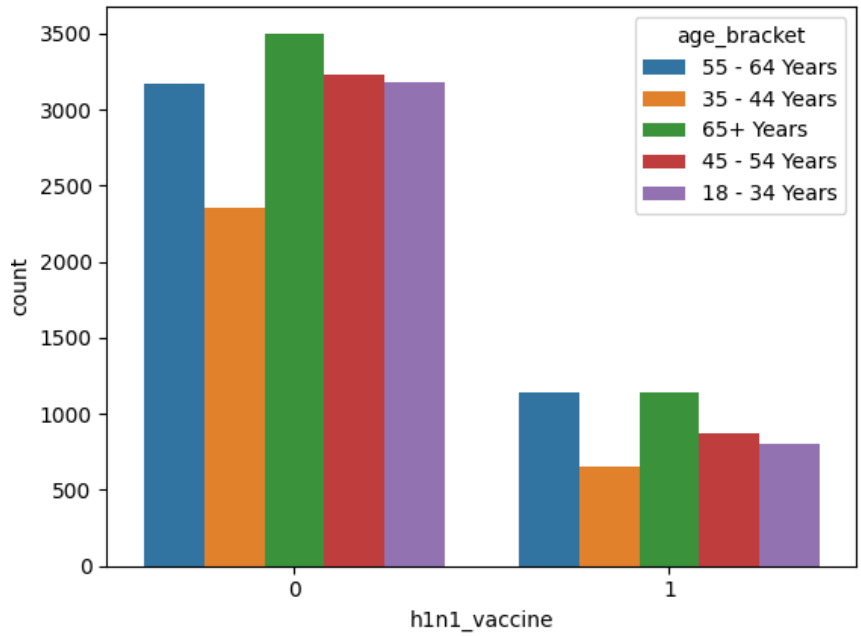
```
In [16]: np.size(np.where(df.h1n1_vaccine==1))
Out[16]: 5674
```

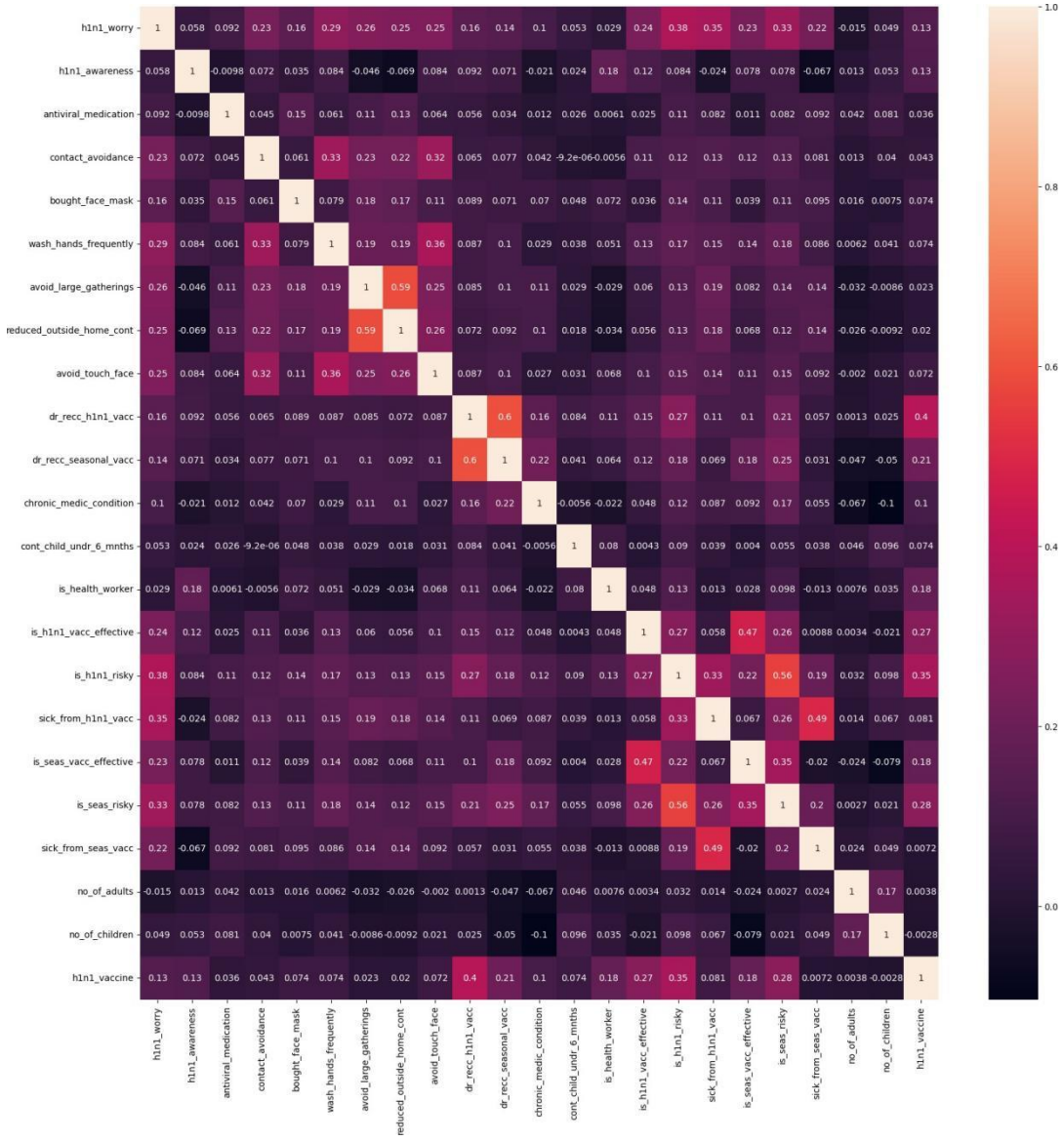
Countplot on number of people got vaccine.

```
In [17]: sns.countplot(x='h1n1_vaccine',data = df)
Out[17]: <Axes: xlabel='h1n1_vaccine', ylabel='count'>
```






age_bracket
55 - 64 Years
35 - 44 Years
65+ Years
45 - 54 Years
18 - 34 Years
count
3500
3000
2500
2000
1500
1000
500
0
0
1
h1n1_vaccine


h1n1_worry
h1n1_awareness
antiviral_medication
contact_avoidance
bought_face_mask
wash_hands_frequently
avoid_large_gatherings
reduced_outside_home_cont
avoid_touch_face
dr_recc_h1n1_vacc
dr_recc_seasonal_vacc
chronic_medic_condition
cont_child_undr_6_mnths
is_health_worker
is_h1n1_vacc_effective
is_h1n1_risky
sick_from_h1n1_vacc
is_seas_vacc_effective
is_seas_risky
sick_from_seas_vacc
no_of_adults
no_of_children
h1n1_vaccine
1.0
0.8
0.6
0.4
0.2
0.0

MODEL TRAINING AND TESTING

In [18]:
```
x,X,y,Y=train_test_split(df.drop(['h1n1_vaccine'],axis=1),df['h1n1_vaccine'],test_size=0.05)
```

In [19]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
```

In [20]:
```
knn.fit(x,y)
```

Out[20]:
```
KNeighborsClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [21]:
```
knn.score(x,y)
```

Out[21]:
```
0.8558984667533799
```

In [22]:
```
knn.score(X,Y)
```

Out[22]:
```
0.8181137724550899
```

- Decision Tree Classifier

In [23]:
```
d_classifier = DecisionTreeClassifier()
d_classifier.fit(x,y)
d_classifier.score(x,y)
```

Out[23]:
```
0.9999211698395806
```

In [24]:
```
d_classifier.score(X,Y)
```

Out[24]:
```
0.7559880239520959
```

- scores:

In [25]:
```
test_score=[]
train_score=[]
for _ in range(1):
    x,X,y,Y=train_test_split(df.drop(['h1n1_vaccine'],axis=1),df['h1n1_vaccine'],test_size=0.05)
    d_classifier.fit(x,y)
    train_score.append(d_classifier.score(x,y))
    test_score.append(d_classifier.score(X,Y))
print("train_socre:",np.mean(train_score),"test_score",np.mean(test_score))
```

```
train_socre: 0.9999605849197903 test_score 0.7402694610778443
```

-

difference between train and test is high so we might not use this decisiontreeclassifier

In [28]:
```
test_score=[]
train_score=[]
for _ in range(1):
    x,X,y,Y=train_test_split(df.drop(['h1n1_vaccine'],axis=1),df['h1n1_vaccine'],test_size=0.05)
    knn.fit(x,y)
    train_score.append(knn.score(x,y))
    test_score.append(knn.score(X,Y))
print("train_socre:",np.mean(train_score),"test_score",np.mean(test_score))
```

```
train_socre: 0.8550707500689764 test_score 0.8016467065868264
```

difference between train and test is less than 5% so we might use this kneighboursclassifier

with accuracy of 80+%

CERTIFICATE:

YHills
Learning Beyond Expectations

INTERNSHIP
COMPLETION CERTIFICATE

OFFICIAL INTERNSHIP PARTNER:

E-Cell
IIT Hyderabad

PROUDLY PRESENTED TO:

PAILA TEJESWARA RAO

has successfully completed 2 months Internship from 01/05/2023 to 30/06/2023 in Data Science at YHills.

Certificate ID : YHI-5003927

15/07/2023
DATE

AMAN KUMAR, CO-FOUNDER

YHills
Learning Beyond Expectations

CERTIFICATE
OF COMPLETION

DATA SCIENCE
CERTIFIED

PROUDLY PRESENTED TO

PAILA TEJESWARA RAO

has successfully completed **DATA SCIENCE** course from 01/05/2023 to 30/06/2023 at YHills.

15/07/2023
DATE

AMAN KUMAR, CO-FOUNDER

Certificate ID : YHIcc-5003927