

# Horology 2.0: Forecasting the Future of Smartwatch Prices using Machine Learning

## 1. INTRODUCTION :

### **1.1.Project Overview:**

Horology 2.0 aims to explore and predict the future trends in smartwatch prices by leveraging data analytics and market insights. The project combines traditional horology principles with modern technology to understand how smartwatch prices evolve over time. Develop predictive models to forecast future smartwatch prices based on identified variables. Analyze historical smartwatch pricing data to identify patterns and trends. Provide insights into potential disruptions or innovations that could influence pricing dynamics.

Incorporate factors such as technological advancements, consumer demand, and market competition into the analysis. Smartwatches have become increasingly popular in recent years due to their ability to provide a range of functionalities beyond traditional timekeeping, such as fitness tracking, communication, and entertainment. Explore and select appropriate machine learning algorithms for regression, considering factors such as model interoperability and prediction accuracy. Here we uses web application.

### **1.2. Purpose:**

In this project, we aim to develop a machine learning model that predicts the price of wristwatch based on a given set of features. Our dataset contains information on various attributes of smartwatches, including brand, model, operating system, connectivity, display type and size, resolution, water resistance, battery life, heart rate monitor, GPS, NFC, and price. We will use this dataset to train and evaluate our predictive model, which has the potential to transform the way consumers and manufacturers approach pricing and purchasing decisions.

Gain a deeper understanding of the factors influencing smartwatch prices over time.Assist smartwatch manufacturers in making strategic decisions related to pricing.Help consumers understand the value proposition of different features in relation to pricing.

## **2. LITERATURE SURVEY :**

### **2.1 Existing Problem :**

The development of smartwatches in the watch industry has been fascinating, cost is still a difficult problem. The pricing of smartwatches is influenced by a number of issues and trends, making this an exciting area to research. This literature review summarizes current issues and future expectations regarding smartwatch pricing:

**Sustainability and Ethical Issues:** The importance of sustainability is becoming when setting prices for products. The increasing awareness of ethical manufacturing practices among consumers may impact their inclination to pay higher prices for eco-friendly smartwatches.

**Consumer Perception and Expectations:** study explores consumer preferences and how much they are willing to pay for particular features on smartwatches. The difficulty is in keeping a competitive price point while matching product offerings to customer expectations.

**Competitive Market Dynamics:** Examines the competitive environment in the smartwatch industry. Companies need to strategically position their offerings to remain competitive without sacrificing quality, as there are many players in the market offering a wide range of products at different price points.

**Costs of Manufacturing and the Supply Chain:** Tells about how difficult it is to control manufacturing and supply chain expenses in the smartwatch sector. Pricing strategies are directly impacted by factors such as economies of scale, assembly costs, and component sourcing.

**Economic Factors and Pricing Strategies:** Analyses highlight how economic factors affect the cost of smartwatches. Pricing strategies are influenced by a number of factors, including global economic conditions, inflation, and currency fluctuations.

**Subscription Models and Service Bundling:** In order to maintain competitive pricing and create recurring revenue streams, some researchers, propose that smartwatch companies investigate subscription-based models or bundle services with hardware.

## 2.2 References:

- [1] Schnabel, P., Hein, D.W., He, J., R, Y.K., Rawhide, S., Krulikowski, B.: Fashion or technology? A fashion perspective on the perception and adoption of augmented reality smart glasses. *i-com J. Interact. Media* 15(2), 179–194 (2016)
- [2] Fang, Y.M., Chang, C.C.: User's psychological perception and perceived readability of wearable devices for elderly people. *Behave. Inf. Techno.* 35(3), 225–232 (2016)
- [3] Chattahoochee, A., Methinks, N.: Re conceptualizing organizational support and its effect on information technology usage: evidence from the healthcare sector. *J. Comput. Inf. Sys t.* 48, 69–76 (2008)
- [4] Venkatesh, V., Thong, J.Y.L., Chan, F.K.Y., Hu, P.J.H., Brown, S.A.: Extending the two-stage information systems continuance model: incorporating UTAUT predictors and the role of context. *Inf. Sys t. J.* 21, 527–555 (2011)
- [5] Ha, T., Beijnon, B., Kim, S., Lee, S., Kim, J.H.: Examining user perceptions of smartwatch through dynamic topic modelling. *Tele mat. Inform.* 34, 1262–1273 (2017)
- [6] Kim, K.J., Shin, D.H.: An acceptance model for smartwatches. *Internet Res.* 25(4), 527–541 (2015)
- [7] Chihuahua, S.H.W., Rauschnabel, P.A., Krey, N., Nguyen, B., Ramayah, T., Lade, S.: Wearable technologies: the role of use-fullness and visibility in smartwatch adoption. *Compute. Hum. Behan.* 65, 276–284 (2016)

## 2.3 Problem Statement Definition:

**Problem Statement:** The smartwatch industry, which is a crucial component of the watch industry, faces a variety of challenges when it comes to pricing strategies because of changing consumer demands, market competition, manufacturing complexities, ethical issues, regulatory requirements, and economic fluctuations. It is imperative that industry stakeholders comprehend these challenges and project the future trajectory of smartwatch prices in the context of Horology 2.0 in order to effectively navigate this dynamic landscape.

**Important Elements in the Problem Statement:**

Provide an overview of the smartwatch industry's current situation in relation to the larger field of horology, highlighting its importance, expansion, and influence on consumer trends.

Difficulties with Pricing Strategies: Draw attention to the difficulties faced by smartwatch producers in setting the prices for their goods, taking into account variables like consumer preferences, manufacturing costs, competitive pressures, ethical issues, legal compliance, and economic swings.

Emphasize the necessity of forecasting smartwatch prices within the context of Horology 2.0 in order to stay competitive in the ever-evolving market, foresee and handle future obstacles, and optimize pricing strategies.

Stakeholder Importance: Explain how this forecasting exercise affects the decisions and industry dynamics of manufacturers, consumers, investors, policymakers, and researchers, and what relevance it has for each of these groups.

The study's objective is to provide industry stakeholders with actionable insights by analyzing the current obstacles to smartwatch pricing and projecting future trends in the same context as Horology 2.0.

Define the study's parameters, including its duration, any intended geographic focus, and the technological, consumer-centric, economic, and other aspects of smartwatch pricing that will be examined.

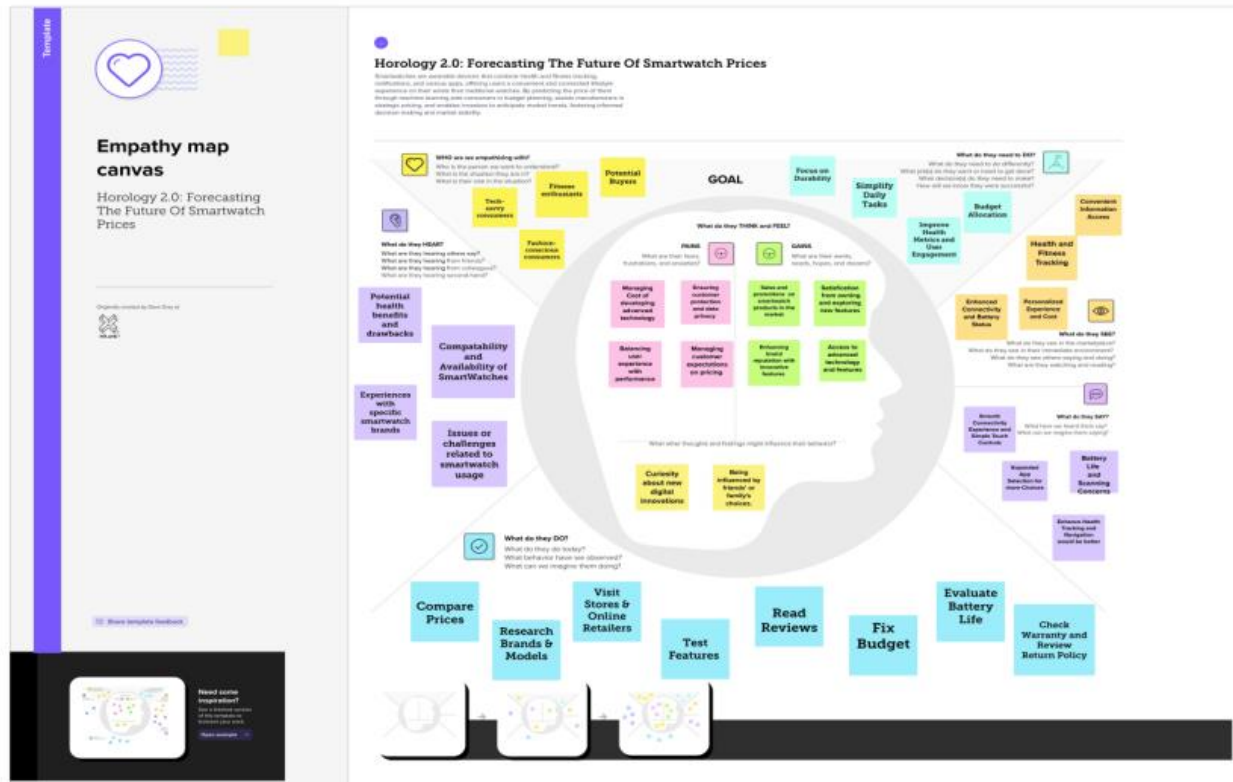
Expected Outcome: Specify the expected deliverables of the research, including suggestions for stakeholders, prospective pricing models, insights into future pricing trends, and approaches to overcoming pricing obstacles.

This problem statement forms the basis for an extensive investigation into the intricacies and anticipated future costs of smartwatches in the dynamic context of Horology 2.0.

### 3. IDEATION & PROPOSED SOLUTION

### 3.1. Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behavior and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.




### 3.2. Ideation & Brainstorming:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

#### Step-1: Team Gathering, Collaboration and Select the Problem Statement

Template



## Brainstorm & idea prioritization

Horology 2.0: Forecasting The Future Of Smartwatch Prices

10 minutes to prepare  
1 hour to collaborate  
2-8 people recommended

1

**Before you collaborate**

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

2

**Team gathering**

Define who should participate in the session and send an invite. Brain research indicates it goes even ahead!

3

**Set the goal**

Think about the problem you'll be focusing on solving in the brainstorming session.

4

**Learn how to use the facilitation tools**

Use the Facilitation Superpowers to set a happy and productive session!

Open article →

1

**Define your problem statement**

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

Problem

How might we develop a machine learning model that accurately predicts smartwatch prices while considering the dynamic interplay of temporal factors, consumer preferences, and complex feature interactions, which collectively impact the pricing of a smartwatch over time?

2

**Key rules of brainstorming**

To run an smooth and productive session

Stay on topic

Encourage wild ideas

Defer judgment

Listen to others

Go for volume

If possible, be visual

Need some inspiration?

Use a random stream of this template to inspire your ideas.

Open resources →

## Step-2: Brainstorm, Idea Listing and Grouping

### Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

**Geetha Pallavi**

- Collect diverse historic smartwatch sales data features.
- Learn Customers likes from their past Interactions
- Create an reliable user interface and simple interaction with smart phone.
- Analyze how smartwatch preferences differ across age, gender and income.

**Divya**

- Analyzing how artificial intelligence and machine learning are becoming integral to smartwatches
- Consider how competition from other wearables affects smartwatch pricing
- Examining how the second hand market for used smartwatches may influence both used and new pricing
- Predicting how smartwatch prices will evolve based on the segmentation of features

**Indhu**

- Incorporating real-time news and event monitoring to adapt price predictions for sudden market shifts, like product launches, recalls, or economic factors.
- Considering seasonal trends and technological advancements for dynamic smartwatch price predictions.
- Considering geographic location as a factor in pricing. Since Smartwatch prices can vary by region due to factors like import/export costs and local demand.
- Integrate sustainability and material analysis data to understand how eco-friendly materials and manufacturing processes impact smartwatch prices.

**Varshini**

- Consider how sustainable materials and manufacturing processes may affect prices in the future
- Explore how regional trends and emerging markets may affect smartwatch prices
- Create a captivating virtual smartwatch game where players can amass a variety of rare smartwatches
- Develop a smartwatch application that combines fitness monitoring and horoscope insights

### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

**Group 1: Community and Social Support Initiatives**

- Analyze how smartwatch preferences differ across age, gender and income.
- Integrate sustainability and material analysis data to understand how eco-friendly materials and manufacturing processes impact smartwatch prices.
- Examining how the second hand market for used smartwatches may influence both used and new pricing
- Explore how regional trends and emerging markets may affect smartwatch prices.

**Group 2: Education, Awareness.**

- Provide an overview of how smartwatches have gained popularity in recent years
- Analyse the current market trends and competition in the smartwatch industry
- Explaining how consumer demand influences pricing
- Provide guidance on how consumers can make informed choices when buying smartwatches

**Group 3: Innovation-Driven Approaches.**

- Create a captivating virtual smartwatch game where players can amass a variety of rare smartwatches
- Develop a smartwatch application that combines fitness monitoring and horoscope insights
- Learn Customers likes from their past Interactions
- Create an reliable user interface and simple interaction with smart phone.

## Step-3: Idea Prioritization

### Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

**Importance**  
Rank of these ideas could get more difficult only when multiple ideas are being ranked based on importance

**Feasibility**  
Ranking of ideas based on how difficult it is to make them happen. Feasibility has a lot of (Cost, time, effort, complexity, etc.)

Examining how the second hand market for used smartwatches may influence both used and new pricing

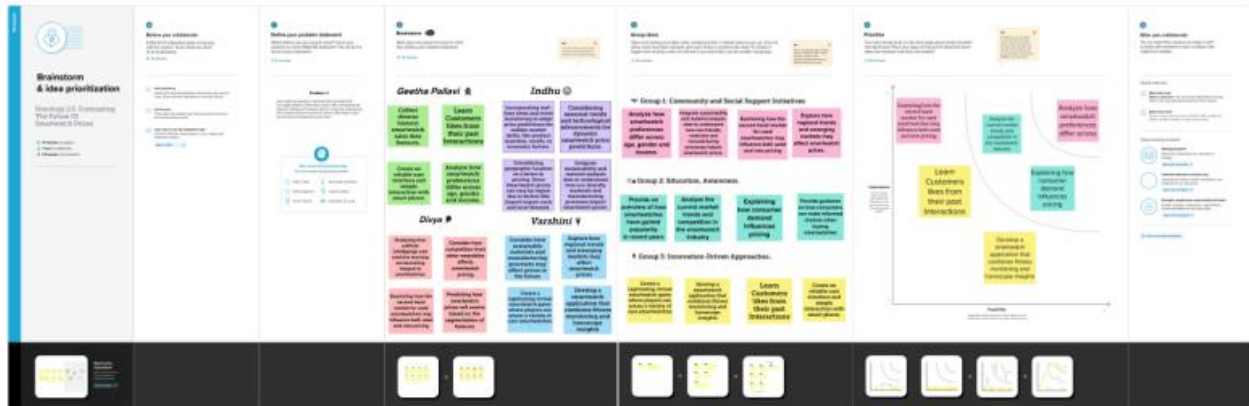
Analyse the current market trends and competition in the smartwatch industry

Analyze how smartwatch preferences differ across

Learn Customers likes from their past Interactions

Explaining how consumer demand influences pricing

Develop a smartwatch application that combines fitness monitoring and horoscope insights



## 4.REQUIREMENT ANALYSIS

### 4.1. Functional requirement:

Functional requirements describe the specific features, capabilities, and behaviors that a system, product, or software application must have to meet the needs of its users and achieve its intended purpose. These requirements focus on what the system should do and are typically expressed in terms of input, process, and output.

User Authentication and Authorization:

Users must be able to register, log in, and securely access the system. Different user roles should have appropriate access levels.

User Profile Management:

Users should be able to create, update, and delete their profiles. Preferences related to smartwatch features and brands should be editable.

Data Collection and Analysis:

The system must collect and analyze historical and real-time data on smartwatch prices. The algorithm should process and interpret the data for accurate forecasting.

Historical Data Archive:

Maintain an archive of historical smartwatch pricing data. Allow users to access and download historical data for analysis.



Notification System:

Implement a notification system to alert users about significant changes or events in the smartwatch market. Allow users to customize notification preferences.

#### **4.2. Non-Functional requirements:**

Non-functional requirements, also known as quality attributes or system qualities, define the characteristics that describe how a system should behave and perform rather than what it should do functionally. These requirements focus on aspects such as performance, security, usability, reliability, and maintainability.

Performance:

The system should respond to user requests within a specified time frame. It must handle a scalable number of users and data points.

Security:

User data should be encrypted and stored securely. The system should protect against unauthorized access and potential cyber threats.

Reliability:

The forecasting algorithm must provide reliable and accurate predictions. The system should have minimal downtime and ensure data integrity.

Scalability:

The system should scale easily to accommodate an increasing number of users and data volume.

Usability:

The user interface should be intuitive and user-friendly. Provide tooltips, help guides, or tutorials to assist users.

Monitoring and Logging:

Implement monitoring tools to track system performance. Maintain logs for troubleshooting and auditing purposes.

## **5. PROJECT DESIGN**

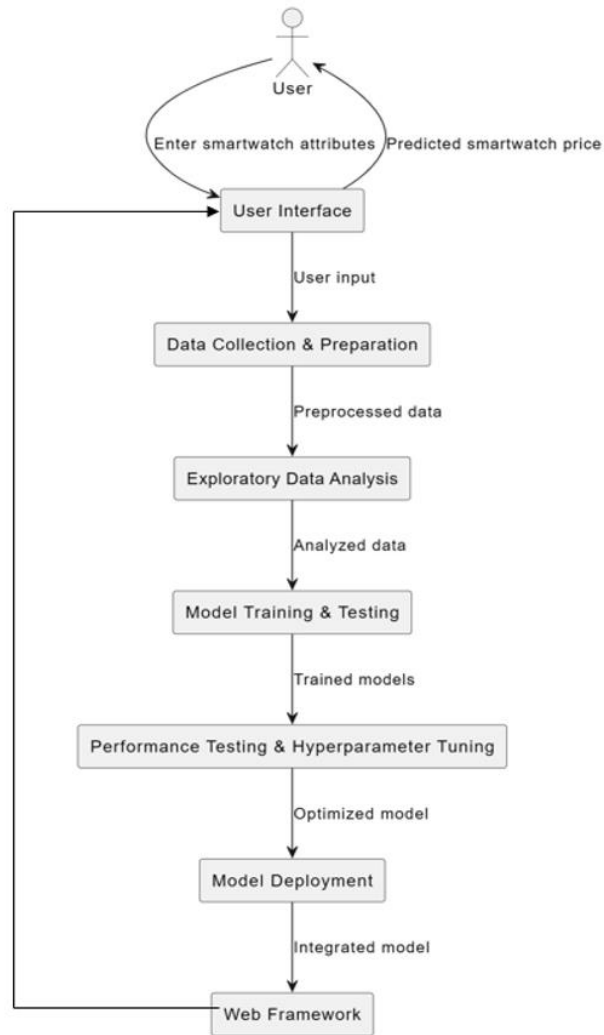
### **5.1 Data Flow Diagrams & User Stories**

#### **Data Flow Diagrams:**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

#### **Data Flow Diagram of Smart Watch Price Prediction Using ML:**

This diagram illustrates the flow of data in the Smartwatch Price Prediction System. The user interacts with the User Interface to input smartwatch attributes. The data flows through several processes, including Data Collection & Preparation, Exploratory Data Analysis, Model Training & Testing, Performance Testing & Hyper parameter Tuning, and Model Deployment. The final result, the predicted smartwatch price, is showcased on the User Interface.



## User Stories :

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Smartwatch manufactures and Retailers	Project Setup & Infrastructure	USD-1	As a smartwatch manufactures, I like to set up the development environment with the required tools and frameworks to start the smartwatch price prediction project.	Successfully configured with all necessary tools and frameworks.	Medium	Sprint-3
Developers	Development Environment	USD-2	As a developer, I like to have use new technologies and develop an application that can help for most of the people.	The development environment should be successfully configured with all necessary tools and frameworks, such as a Python interpreter, machine learning libraries, and data visualization tools.	Medium	Sprint-4
Researches	Data Collection	USD-3	As a researcher, I like to gather a diverse dataset of smartwatch prices and features from various sources (e.g., online retailers, manufacturer websites, etc.) for training the machine learning model.	The dataset should be diverse and representative of the smartwatch market, with a wide range of prices and features. It should also be cleaned and preprocessed to ensure that it is suitable for machine learning.	High	Sprint-1
Data Scientists	Data Preprocessing	USD-4	As a data scientist, I like to preprocess the collected dataset by cleaning and normalizing the data, and splitting it into training and validation sets.	The dataset should be cleaned and normalized to ensure that it is consistent and easy for the machine learning model to understand. It should also be split into training and validation sets so that the model's performance can be evaluated.	High	Sprint-2
Machine learning Engineers	Model development	USD-5	As a Machine Learning Engineer, I like to explore and evaluate different machine learning algorithms (e.g., linear regression, decision trees, random forests, etc.) to select the most suitable model for smartwatch price prediction.	A variety of machine learning algorithms should be explored and evaluated to select the one that performs best on the smartwatch price	Medium	Sprint-2

				prediction task. This may involve tuning the hyperparameters of each algorithm to optimize its performance.		
Tech Enthusiasts	Training	USD-6	As a Tech Enthusiasts, I like to train the selected machine learning model using the preprocessed dataset and monitor its performance on the validation set. As a tech enthusiast, I want to be able to learn about the different features of smartwatches and how they affect their price, so that I can make an informed decision about which smartwatch to buy.	The selected machine learning model should be trained on the preprocessed dataset. The model's performance should be monitored on the validation set to ensure that it is generalizing well and is not overfitting to the training data.	High	Sprint-1
Outdoor Adventurers	Model deployment & Integration	USD-7	As an Outdoor Adventurers, I like to deploy the trained machine learning model as an API or web service to make it accessible for smartwatch price prediction. As an outdoor adventurer, I want to be able to see reviews of smartwatches before I buy one, so that I can get an idea of what other people think of the smartwatch and its features, especially features that are important to outdoor adventurers.	The trained machine learning model should be deployed as an API or web service so that it can be used by others to predict smartwatch prices. This may involve using a cloud computing platform, such as Google Cloud Platform or Amazon Web Services.	Medium	Sprint-2
Students	Testing & quality assurance	USD-8	As a student I like to develop, a user-friendly interface to allow users to interact with the machine learning model and get smartwatch price predictions. As a student, I want to be able to use a smartwatch price prediction model to estimate the price of a smartwatch before I buy it, so that I can make an informed decision	The trained machine learning model should be deployed as an API or web service so that it can be used by others to predict smartwatch prices. This may involve using a cloud computing platform, such as Google Cloud Platform or Amazon Web Services.	Medium	Sprint-3
Business Professionals	User input and prediction	USD-9	As a Business professionals, I like to give the Specific features as input and predict the cost of the smart watch. As a business professional, I want to be able to receive notifications when the smartwatch price prediction model is updated with new data or feedback.	A user-friendly interface should be developed to allow users to interact with the machine learning model and get smartwatch price predictions. This	High	Sprint-4
				interface may be a web application.		

## 5.2. Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

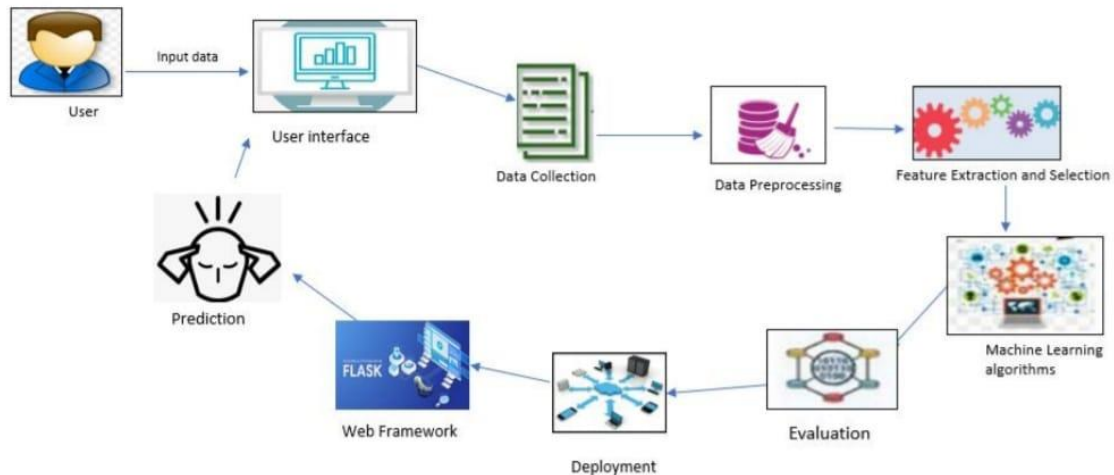
- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.

- Provide specifications according to which the solution is defined, managed, and delivered.

**Model:**

Horology 2.0 envisions a future where smartwatch prices are intelligently forecasted using machine learning models. Leveraging historical sales data, user preferences, market trends, and economic indicators, these models will employ advanced predictive analytics to offer dynamic pricing strategies. By continuously monitoring supply and demand fluctuations, user reviews, and technological advancements, smartwatch manufacturers can optimize their pricing, ensuring competitive and fair rates. Additionally, personalized pricing models can be developed, tailoring offers to individual consumers based on their preferences and usage patterns. Machine learning algorithms will empower the industry to strike a balance between affordability and cutting-edge technology, ensuring that smartwatches remain accessible and appealing to a broad audience while adapting to the ever-evolving wearables market.

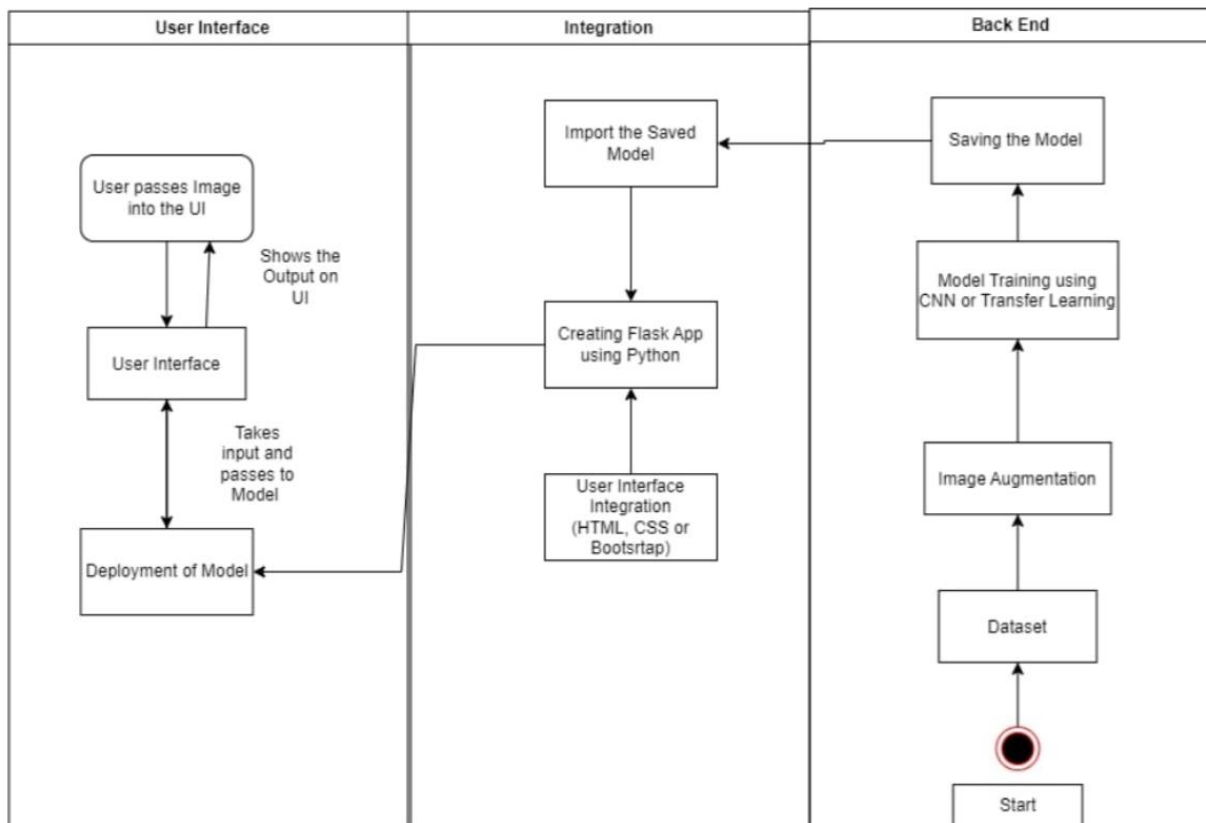
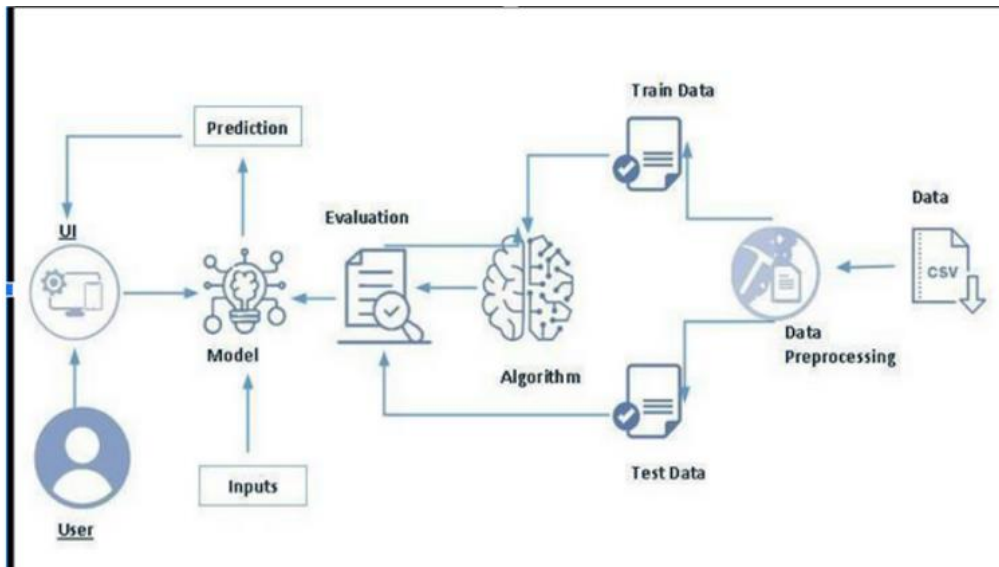
1. Data Collection
2. Data Preprocessing
3. Feature Extraction
4. Machine Learning Algorithms
5. Model Evaluation
6. Deployment
7. Monitoring and Maintenance
8. Insights and Reporting
9. Security and Compliance
10. Scalability



## **6. PROJECT PLANNING & SCHEDULING**

### **6.1. Technical Architecture:**

Technical architecture refers to the structure and organization of a system's hardware and software components, as well as the way they interact to fulfill a set of requirements. It provides a blueprint for the design and implementation of a technology solution, outlining the arrangement and relationships among various elements to achieve specific goals.





## 6.2. Sprint Planning & Estimation:

Sprint Planning and Estimation are key activities in Agile methodologies, particularly in Scrum, a popular Agile framework. These activities help teams organize their work, set priorities, and determine how much work can be accomplished in a specific time frame, known as a sprint.

### Sprint Planning

Establish the goals and scope of the upcoming sprint. Decide which items from the product backlog will be included in the sprint backlog.

### Estimation

Determine the effort required to complete each backlog item. Facilitate better planning and resource allocation.

Sprint	User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Story Points	Priority	Team Members
Sprint-3	Smartwatch manufactures and Retailers	Project Setup & Infrastructure	USD-1	As a smartwatch manufacturer, I like to set up the development environment with the required tools and frameworks to start the smartwatch price prediction project.	Successfully configured with all necessary tools and frameworks.	2	Medium	Pallavi
Sprint-4	Developers	Development Environment	USD-2	As a developer, I like to use new technologies and develop an application that can help for most of the people.	The development environment should be successfully configured with all necessary tools and frameworks, such as a Python interpreter, machine learning libraries, and data visualization tools.	2	Medium	Varshini
Sprint-1	Researches	Data Collection	USD-3	As a researcher, I like to gather a diverse dataset of smartwatch prices and features from various sources (e.g., online retailers, manufacturer websites, etc.) for training the machine learning model.	The dataset should be diverse and representative of the smartwatch market, with a wide range of prices and features. It should also be cleaned and preprocessed to ensure that it is suitable for machine learning.	2	High	Divya Varshini Indhu Pallavi
Sprint-2	Data Scientists	Data Preprocessing	USD-4	As a data scientist, I like to preprocess the collected dataset by cleaning and normalizing the data and splitting it into training and validation sets.	The dataset should be cleaned and normalized to ensure that it is consistent and easy for the machine-learning model to understand. It should also be split into training and validation sets so that the model's performance can be evaluated.	3	High	Indhu

Sprint-2	Machine learning Engineers	Model Development	USD-5	As a Machine Learning Engineer, I like to explore and evaluate different machine learning algorithms (e.g., linear regression, decision trees, random forests, etc.) to select the most suitable model for smartwatch price prediction.	A variety of machine learning algorithms should be explored and evaluated to select the one that performs best on the smartwatch price prediction task. This may involve tuning the hyperparameters of each algorithm to optimize its performance.	3	Medium	Divya
Sprint-1	Tech Enthusiasts	Training	USD-6	As a Tech Enthusiast, I like to train the selected machine learning model using the preprocessed dataset and monitor its performance on the validation set. As a tech enthusiast, I want to be able to learn about the different features of smartwatches and how they affect their price so that I can make an informed decision about which smartwatch to buy.	The selected machine learning model should be trained on the preprocessed dataset. The model's performance should be monitored on the validation set to ensure that it is generalizing well and is not overfitting to the training data.	8	High	Varshini
Sprint-2	Outdoor Adventurers	Model Development and Integration	USD-7	As an Outdoor adventurer, I like to deploy the trained machine learning model as an API or web service to make it accessible for smartwatch price prediction. As an outdoor adventurer, I want to be able to see reviews of smartwatches before I buy one so that I can get an idea of what other people think of the smartwatch and its features, especially features that are important to outdoor adventurers.	The trained machine learning model should be deployed as an API or web service so that it can be used by others to predict smartwatch prices. This may involve using a cloud computing platform, such as Google Cloud Platform or Amazon Web Services.	4	Medium	Pallavi
Sprint-3	Students	Testing and Quality Assurance	USD-8	As a student I like to develop, a user-friendly interface to allow users to interact with the machine learning model and get smartwatch price predictions. As a student, I want to be able to use a smartwatch price prediction model to estimate the price of a smartwatch before I buy it so that	The trained machine learning model should be deployed as an API or web service so that it can be used by others to predict smartwatch prices. This may involve using a cloud computing platform, such as Google Cloud Platform or Amazon Web Services.	3	Medium	Divya

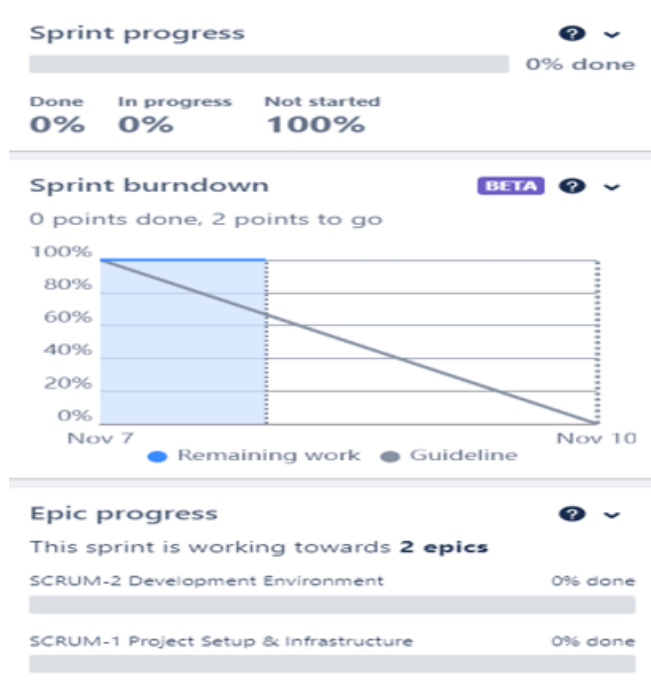
				I can make an informed decision				
Sprint-4	Business Professionals	User Input and Prediction	USD-9	As a Business professional, I like to give Specific features as input and predict the cost of the smartwatch. As a business professional, I want to be able to receive notifications when the smartwatch price prediction model is updated with new data or feedback.	A user-friendly interface should be developed to allow users to interact with the machine-learning model and get smartwatch price predictions. This interface may be a web application.	3	High	Indhu

### 6.3. Sprint Delivery Schedule:

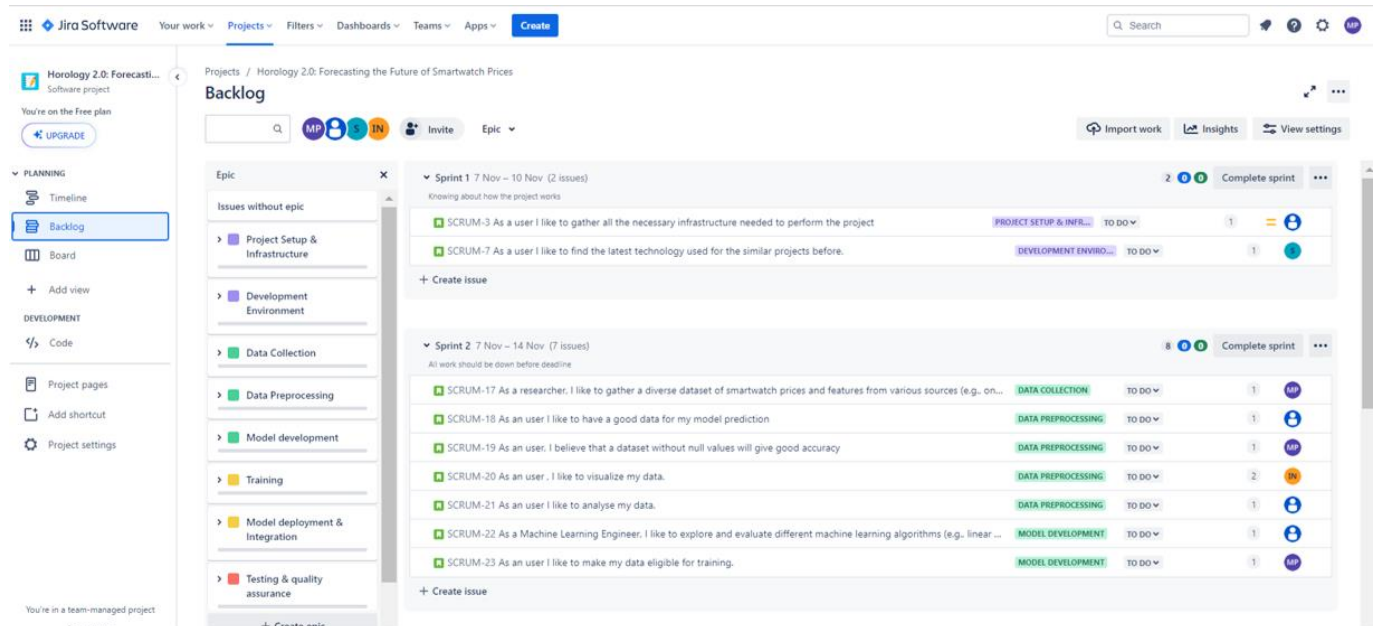
In Scrum, a sprint is a time-boxed iteration during which a potentially shippable product increment is created. Sprints are usually two to four weeks long, and they provide a consistent, short timeframe for development teams to deliver incremental value to the product.

## Burndown Chart:

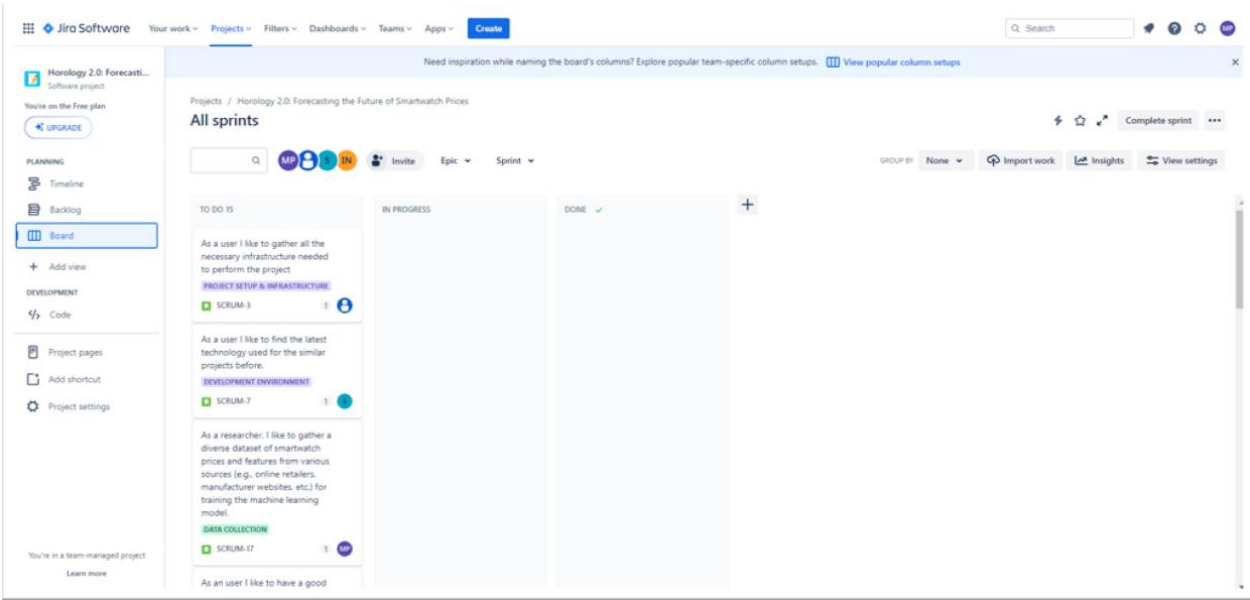
A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



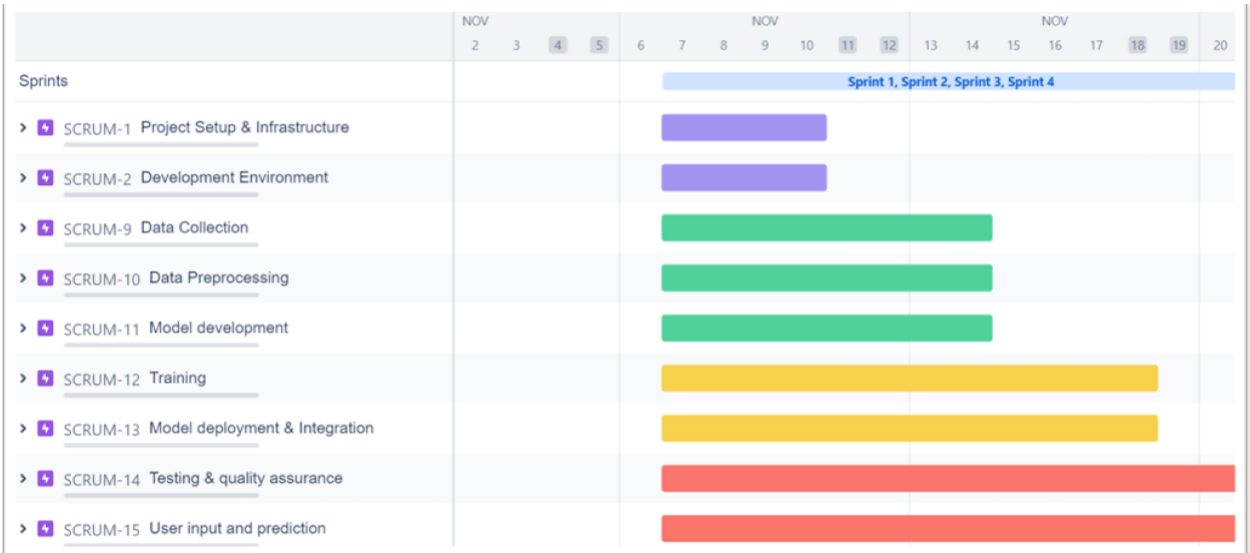
## Backlog Chart:



Board chart:



Time line chart:



## **7. CODING & SOLUTIONING**

### **7.1 Feature 1:**

#### Advanced Health Monitoring:

Integration of more advanced health monitoring features, such as continuous glucose monitoring, advanced sleep tracking, and stress level monitoring.

#### Modular Design:

Smartwatches with modular components, allowing users to upgrade certain features or replace specific parts without having to buy an entirely new device.

#### Extended Battery Life:

Improvements in battery technology to extend the overall battery life of smartwatches, reducing the need for frequent charging.

#### Enhanced Connectivity:

Improved connectivity options, such as 5G support, to enable faster data transfer and more seamless integration with other smart devices.

#### Customization Options:

Greater customization options for users, including the ability to personalize watch faces, straps, and other aesthetic elements.

### **7.2 Feature 2:**

#### AR and VR Integration:

Integration of augmented reality (AR) and virtual reality (VR) features, enhancing user experiences beyond traditional smartwatch functionalities.

#### Eco-Friendly Materials:

Use of sustainable and eco-friendly materials in the construction of smartwatches, aligning with a growing emphasis on environmental sustainability.

Artificial Intelligence (AI) Integration:

Smarter AI algorithms for more personalized and context-aware interactions, providing users with tailored information and recommendations.

Enhanced Display Technologies:

Adoption of advanced display technologies, such as flexible or foldable screens, providing larger and more versatile display options.

Security and Privacy Features:

Enhanced security measures, including biometric authentication methods and improved data privacy controls, to address growing concerns about personal information protection.

### 7.3 Database Scheme:

#### 1: Collect the dataset

There are many popular open sources for collecting the data. E.g., kaggle.com, UCI repository, etc. In this project we have used .csv data. This data is downloaded from kaggle.com.

#### Import necessary libraries

Import the libraries required for this machine learning project

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
```

#### Import the dataset

Our dataset format could be in .csv, excel files,.txt, .json, and so on. With the help of pandas, we can read the dataset. Since our dataset is a csv file, we use read\_csv() which is pandas function to read the dataset. As a parameter we have to give the directory of the csv file. df.head() will display first 5 rows of the dataset.

```
df=pd.read_csv("Smart watch prices.csv")
```

```
df.head()
```

	Brand	Model	Operating System	Connectivity	Display Type	Display Size (inches)	Resolution	Water Resistance (meters)	Battery Life	Heart Rate Monitor	GPS	NFC	Price (USD)
0	Apple	Watch Series 7	watchOS	Bluetooth, Wi-Fi, Cellular	Retina	1.90	396 x 484	50	18	Yes	Yes	Yes	\$399
1	Samsung	Galaxy Watch 4	Wear OS	Bluetooth, Wi-Fi, Cellular	AMOLED	1.40	450 x 450	50	40	Yes	Yes	Yes	\$249
2	Garmin	Venu 2	Garmin OS	Bluetooth, Wi-Fi	AMOLED	1.30	416 x 416	50	11	Yes	Yes	No	\$399
3	Fitbit	Versa 3	Fitbit OS	Bluetooth, Wi-Fi	AMOLED	1.58	336 x 336	50	6	Yes	Yes	Yes	\$229
4	Fossil	Gen 6	Wear OS	Bluetooth, Wi-Fi	AMOLED	1.28	416 x 416	30	24	Yes	Yes	Yes	\$299

## Data Preparation

Data preparation, also known as data preprocessing, is the process of cleaning, transforming, and organizing raw data before it can be used in a data analysis or machine learning model. The activity include following steps:

removing missing values

handling outliers

encoding categorical variables

normalizing data

## Handling Missing Values

Let's first figure out what kind of data is in our columns by using `df.info()`. We may deduce from this that our columns include data of the types "object" and "float64"

```
# Getting the information of data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 379 entries, 0 to 378
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Brand                                378 non-null    object
1   Model                                378 non-null    object
2   Operating System                     376 non-null    object
3   Connectivity                          378 non-null    object
4   Display Type                         377 non-null    object
5   Display Size (inches)                376 non-null    float64
6   Resolution                           375 non-null    object
7   Water Resistance (meters)             378 non-null    object
8   Battery Life                         378 non-null    object
9   Heart Rate Monitor                   378 non-null    object
10  GPS                                  378 non-null    object
11  NFC                                  378 non-null    object
12  Price (USD)                          378 non-null    object
dtypes: float64(1), object(12)
memory usage: 38.6+ KB
```

```
# Checking for the null values
df.isnull().any()
```

```
Brand      True
Model      True
Operating System  True
Connectivity  True
Display Type  True
Display Size (inches)  True
Resolution  True
Water Resistance (meters)  True
Battery Life  True
Heart Rate Monitor  True
GPS         True
NFC         True
Price (USD)  True
dtype: bool
```

```
df.isnull().sum()
```

```
Brand      1
Model      1
Operating System  3
Connectivity  1
Display Type  2
Display Size (inches)  3
Resolution  4
Water Resistance (meters)  1
Battery Life  1
Heart Rate Monitor  1
GPS         1
NFC         1
Price (USD)  1
dtype: int64
```



```
# Replacing Null values in Cateogrical column with mode.
object_columns=df.select_dtypes(include=["object"]).columns
for col in object_columns:
    mode_value=df[col].mode()[0]
    df[col]=df[col].fillna(mode_value)
```

```
# Replacing Null values in float columns with mean
float_columns= df.select_dtypes(include=["float"]).columns
for col in float_columns:
    mean_value=df[col].mean()
    df[col]=df[col].fillna(mean_value)
```

```
df.isnull().any()
```

```
Brand                False
Model                False
Operating System     False
Connectivity         False
Display Type         False
Display Size (inches) False
Resolution           False
Water Resistance (meters) False
Battery Life         False
Heart Rate Monitor   False
GPS                 False
NFC                 False
Price (USD)          False
dtype: bool
```

## Handling Independent Columns

We can use the `df.rename()` function to rename specific columns in the dataset for simplicity.

```
# Handling Independent columns
```

```
# We are renaming the columns for easy coding.
```

```
df=df.rename(columns={'Display Size (inches)': 'Display Size', 'Water Resistance (meters)': 'Water Resistance', 'Battery Life (days)': 'Battery Life', 'Price (USD)': 'Price'})
```

```
df.head()
```

	Brand	Model	Operating System	Connectivity	Display Type	Display Size	Resolution	Water Resistance	Battery Life	Heart Rate Monitor	GPS	NFC	Price
0	Apple	Watch Series 7	watchOS	Bluetooth, Wi-Fi, Cellular	Retina	1.90	396 x 484	50	18	Yes	Yes	Yes	\$399
1	Samsung	Galaxy Watch 4	Wear OS	Bluetooth, Wi-Fi, Cellular	AMOLED	1.40	450 x 450	50	40	Yes	Yes	Yes	\$249
2	Garmin	Venu 2	Garmin OS	Bluetooth, Wi-Fi	AMOLED	1.30	416 x 416	50	11	Yes	Yes	No	\$399
3	Fitbit	Versa 3	Fitbit OS	Bluetooth, Wi-Fi	AMOLED	1.58	336 x 336	50	6	Yes	Yes	Yes	\$229
4	Fossil	Gen 6	Wear OS	Bluetooth, Wi-Fi	AMOLED	1.28	416 x 416	30	24	Yes	Yes	Yes	\$299

The first line of code provides a list of all unique values in the 'Water Resistance' column of the dataframe. The second line of code provides descriptive statistics for the 'Water Resistance' column in order to determine the most frequently observed value. The third line of code changes

all instances of 'Not specified' in the 'Water Resistance' column with '50'. When dealing with missing or confusing data, this is useful.

```
df['Water Resistance'].unique()
```

```
array(['50', '30', '100', '1.5', 'Not specified', '200', '10'],  
      dtype=object)
```

```
df['Water Resistance'].describe()
```

```
count      379  
unique       7  
top         50  
freq       276  
Name: Water Resistance, dtype: object
```

```
# We are replacing the not specified values in water resistance column  
df['Water Resistance']=df['Water Resistance'].replace({'Not specified':'50'})
```

```
df['Display Size'].unique()
```

```
array([1.9      , 1.4      , 1.3      , 1.58      , 1.28      ,  
       1.43      , 1.75      , 1.39      , 1.36316489, 1.65      ,  
       1.2       , 1.57      , 1.       , 1.78      , 1.91      ,  
       1.38      , 1.06      , 1.35      , 1.34      , 0.9       ,  
       1.04      , 1.64      , 1.19      , 4.01      , 1.6       ,  
       1.42      , 2.1       , 1.23      , 1.1       , 1.22      ,  
       1.5       , 1.36      , 1.32      ,  ])
```

```
# Rounding the above float numbers to 1 decimal point.  
df['Display Size']=df['Display Size'].round(1)
```

```
df['Battery Life'].unique()
```

```
array(['18', '40', '11', '6', '24', '14', '2', '4', '12', '30', '3', '45',  
       '5', '10', '48', '7', '16', '9', '25', '72', '60', '56', '70', '1',  
       '48 hours', '15', 'Unlimited', '1.5', '20', '8'], dtype=object)
```

```
df['Battery Life'].describe()
```

```
count      379  
unique      30  
top         14  
freq        84  
Name: Battery Life, dtype: object
```

The first line of code retrieves all the unique values in the "Display Size" column of the dataframe. The second line of code rounds the values in the "Display Size" column to one decimal place. This is done to simplify and make the data more consistent.

```
# We replace all the 48 hours and the Unlimited with top value of battery Life describe to deal with missing and confusing data.  
df['Battery Life']=df['Battery Life'].replace({'48 hours':'14','Unlimited':'14'})
```

```
# It removes the dollar symbol before the price.  
df['Price']=df['Price'].str[1:]
```

```
df['Water Resistance']=df['Water Resistance'].astype(float)  
df['Battery Life']=df['Battery Life'].astype(float)  
df['Price']=df['Price'].str.replace('$','').astype(float)
```

## Handling Categorical Values

In this code, the LabelEncoder object is first created. Then, the categorical columns in the dataset are identified and stored in a list. The for loop is then used to iterate through each categorical column, and the fit\_transform method of the LabelEncoder object is applied to encode the categorical values in each column with unique integer values.

```
# Label Encoding
```

```
from sklearn.preprocessing import LabelEncoder  
lb = LabelEncoder()
```

```
categorical_cols = df.select_dtypes(include=['object']).columns  
for col in categorical_cols:  
    df[col] = lb.fit_transform(df[col])
```

```
df.head()
```

	Brand	Model	Operating System	Connectivity	Display Type	Display Size	Resolution	Water Resistance	Battery Life	Heart Rate Monitor	GPS	NFC	Price
0	1	127	34	2	17	1.9	27	50.0	18.0	0	1	1	399.0
1	30	36	31	2	0	1.4	31	50.0	40.0	0	1	1	249.0
2	8	105	9	1	0	1.3	30	50.0	11.0	0	1	0	399.0
3	6	109	7	1	0	1.6	19	50.0	6.0	0	1	1	229.0
4	7	43	31	1	0	1.3	30	30.0	24.0	0	1	1	299.0

```
df = df.drop('Heart Rate Monitor', axis=1)
```

```
df.head()
```

	Brand	Model	Operating System	Connectivity	Display Type	Display Size	Resolution	Water Resistance	Battery Life	GPS	NFC	Price
0	1	127	34	2	17	1.9	27	50.0	18.0	1	1	399.0
1	30	36	31	2	0	1.4	31	50.0	40.0	1	1	249.0
2	8	105	9	1	0	1.3	30	50.0	11.0	1	0	399.0
3	6	109	7	1	0	1.6	19	50.0	6.0	1	1	229.0
4	7	43	31	1	0	1.3	30	30.0	24.0	1	1	299.0

```
- - - - -
```

## Data Visualization

The purpose of descriptive analysis is to analyze the basic features of data using a statistical technique. In this case, Pandas has a useful function called describe. We can understand the

unique, top, and frequent values of categorical features with this describe function. We can also get the count, mean, standard deviation, minimum, maximum, and percentile values of continuous features

```
df.describe(include='all')
```

	Brand	Model	Operating System	Connectivity	Display Type	Display Size	Resolution	Water Resistance	Battery Life	GPS	NFC	Price
count	379.000000	379.000000	379.000000	379.000000	379.000000	379.000000	379.000000	379.000000	379.000000	379.000000	379.000000	379.000000
mean	18.168865	68.606860	20.778364	1.203166	6.941953	1.368074	22.139842	52.804749	12.208443	0.920844	0.83905	312.910290
std	13.040757	38.933753	11.407946	0.532927	8.978918	0.219087	9.080415	26.939235	12.326042	0.270338	0.36797	202.163738
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.900000	0.000000	1.500000	1.000000	0.000000	0.00000	49.000000
25%	7.000000	33.500000	9.000000	1.000000	0.000000	1.200000	17.500000	50.000000	3.000000	1.000000	1.00000	199.000000
50%	16.000000	71.000000	27.000000	1.000000	0.000000	1.400000	23.000000	50.000000	11.000000	1.000000	1.00000	279.000000
75%	31.000000	102.000000	31.000000	1.000000	14.000000	1.400000	32.000000	50.000000	15.000000	1.000000	1.00000	329.000000
max	41.000000	136.000000	34.000000	4.000000	26.000000	4.000000	35.000000	200.000000	72.000000	1.000000	1.00000	1800.000000

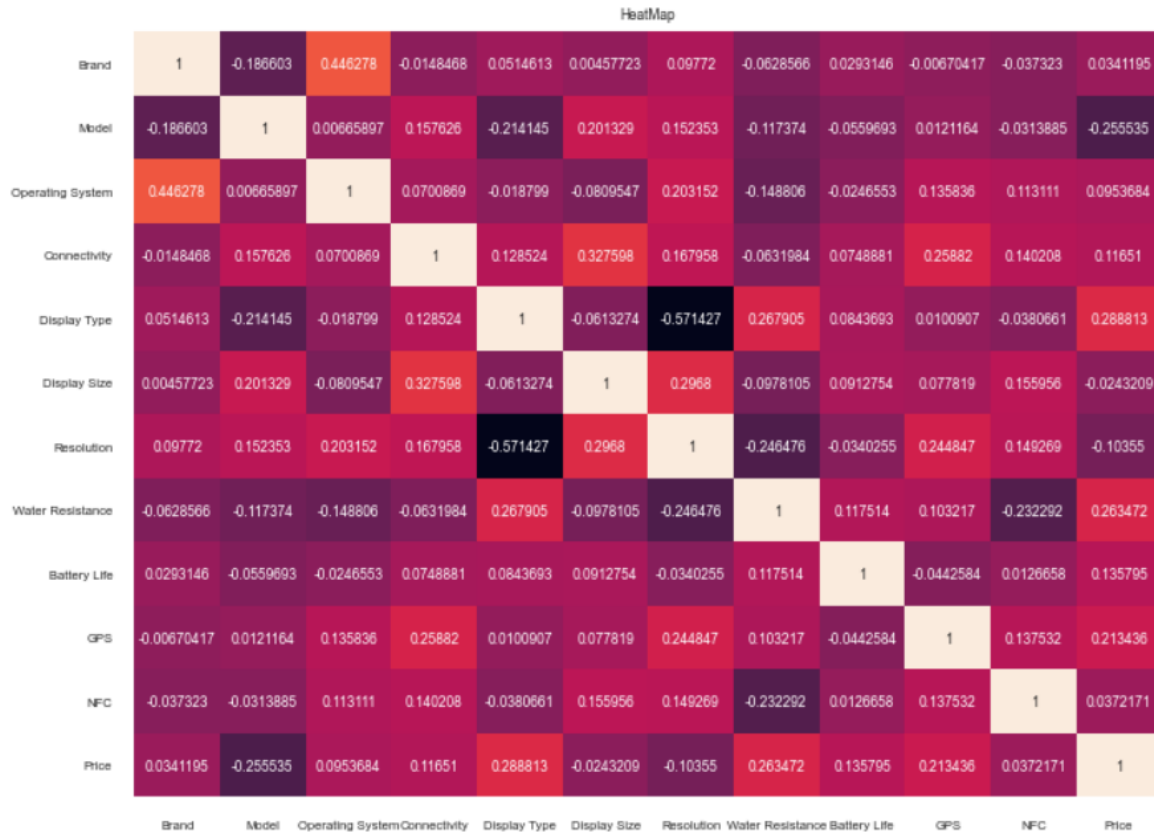
```
corr= df.corr()
```

```
corr
```

	Brand	Model	Operating System	Connectivity	Display Type	Display Size	Resolution	Water Resistance	Battery Life	GPS	NFC	Price
Brand	1.000000	-0.186603	0.446278	-0.014847	0.051461	0.004577	0.097720	-0.062857	0.029315	-0.006704	-0.037323	0.034120
Model	-0.186603	1.000000	0.006659	0.157626	-0.214145	0.201329	0.152353	-0.117374	-0.055969	0.012116	-0.031389	-0.255535
Operating System	0.446278	0.006659	1.000000	0.070087	-0.018799	-0.080955	0.203152	-0.148806	-0.024655	0.135836	0.113111	0.095368
Connectivity	-0.014847	0.157626	0.070087	1.000000	0.128524	0.327598	0.167958	-0.063198	0.074888	0.258820	0.140208	0.116510
Display Type	0.051461	-0.214145	-0.018799	0.128524	1.000000	-0.061327	-0.571427	0.267905	0.084369	0.010091	-0.038066	0.288813
Display Size	0.004577	0.201329	-0.080955	0.327598	-0.061327	1.000000	0.296800	-0.097811	0.091275	0.077819	0.155956	-0.024321
Resolution	0.097720	0.152353	0.203152	0.167958	-0.571427	0.296800	1.000000	-0.246476	-0.034026	0.244847	0.149269	-0.103550
Water Resistance	-0.062857	-0.117374	-0.148806	-0.063198	0.267905	-0.097811	-0.246476	1.000000	0.117514	0.103217	-0.232292	0.263472
Battery Life	0.029315	-0.055969	-0.024655	0.074888	0.084369	0.091275	-0.034026	0.117514	1.000000	-0.044258	0.012666	0.135795
GPS	-0.006704	0.012116	0.135836	0.258820	0.010091	0.077819	0.244847	0.103217	-0.044258	1.000000	0.137532	0.213436
NFC	-0.037323	-0.031389	0.113111	0.140208	-0.038066	0.155956	0.149269	-0.232292	0.012666	0.137532	1.000000	0.037217
Price	0.034120	-0.255535	0.095368	0.116510	0.288813	-0.024321	-0.103550	0.263472	0.135795	0.213436	0.037217	1.000000

```
sns.set(font_scale=0.5) # Set font scale to increase the font size in the heatmap
plt.figure(figsize=(8, 6)) # Set the figure size to increase the size of the heatmap
sns.heatmap(corr, annot=True, fmt='g', cbar=False)
plt.title('HeatMap')
```

Text(0.5, 1.0, 'HeatMap')



```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x2995be1ff90>
```



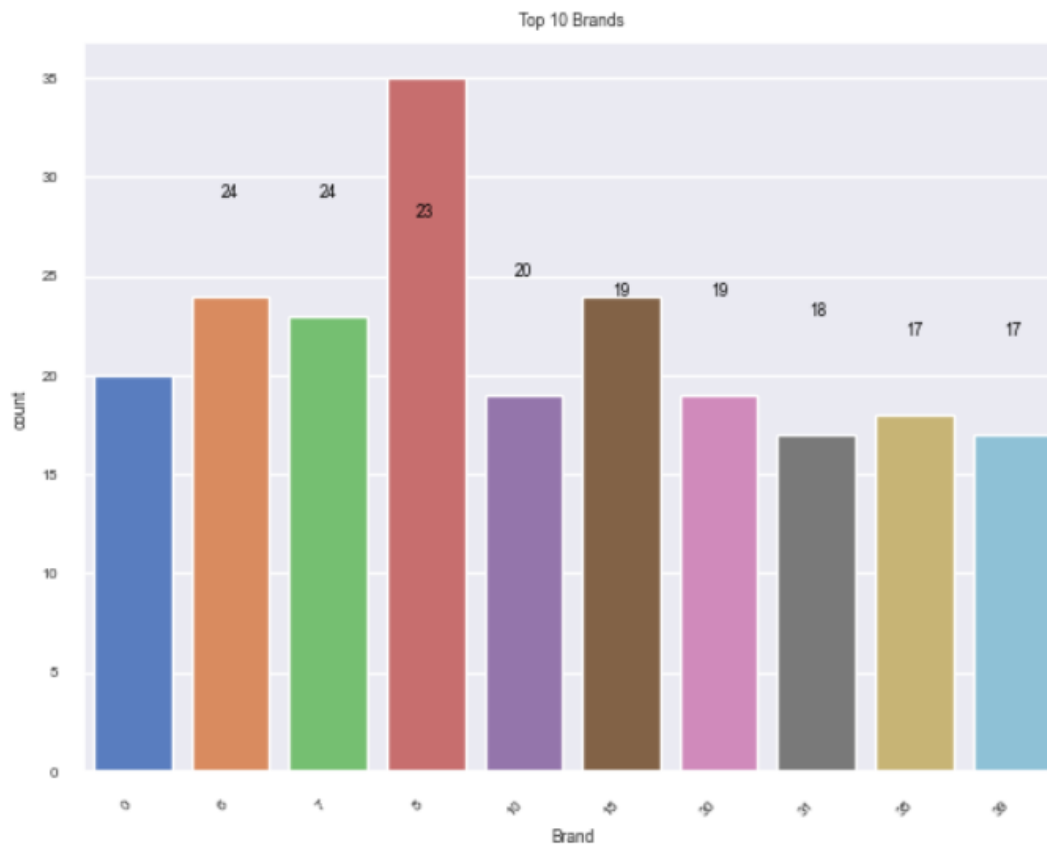
## Univariate analysis:

Univariate analysis is a statistical method used to analyse a single variable in a dataset. This analysis focuses on understanding the distribution, central tendency, and dispersion of a single variable. The code below creates a bar plot using the Seaborn library to show the top 10 brands in a dataset, along with their respective counts. It first gets the top 10 brands and their counts using the `value_counts()` method, then sets the plot style, creates the bar plot using the `barplot()` method, and rotates the x-tick labels for better visibility. Value labels are added on the bars using the `text()` method, and the axis labels and title are set using the `set()` method. Finally, the plot is displayed using the `show()` method

```

top_brands= df['Brand'].value_counts().index[:10]
counts = df['Brand'].value_counts().values[:10]
sns.set_style("darkgrid")
ax = sns.barplot(x=top_brands, y=counts, palette="muted")
ax.set_xticklabels (ax.get_xticklabels (), rotation=45, ha='right')
for i, v in enumerate(counts):
    ax.text(i, v+5, str(v), color='black', ha='center')
ax.set(xlabel='Brand', ylabel='count', title='Top 10 Brands')
plt.show()

```

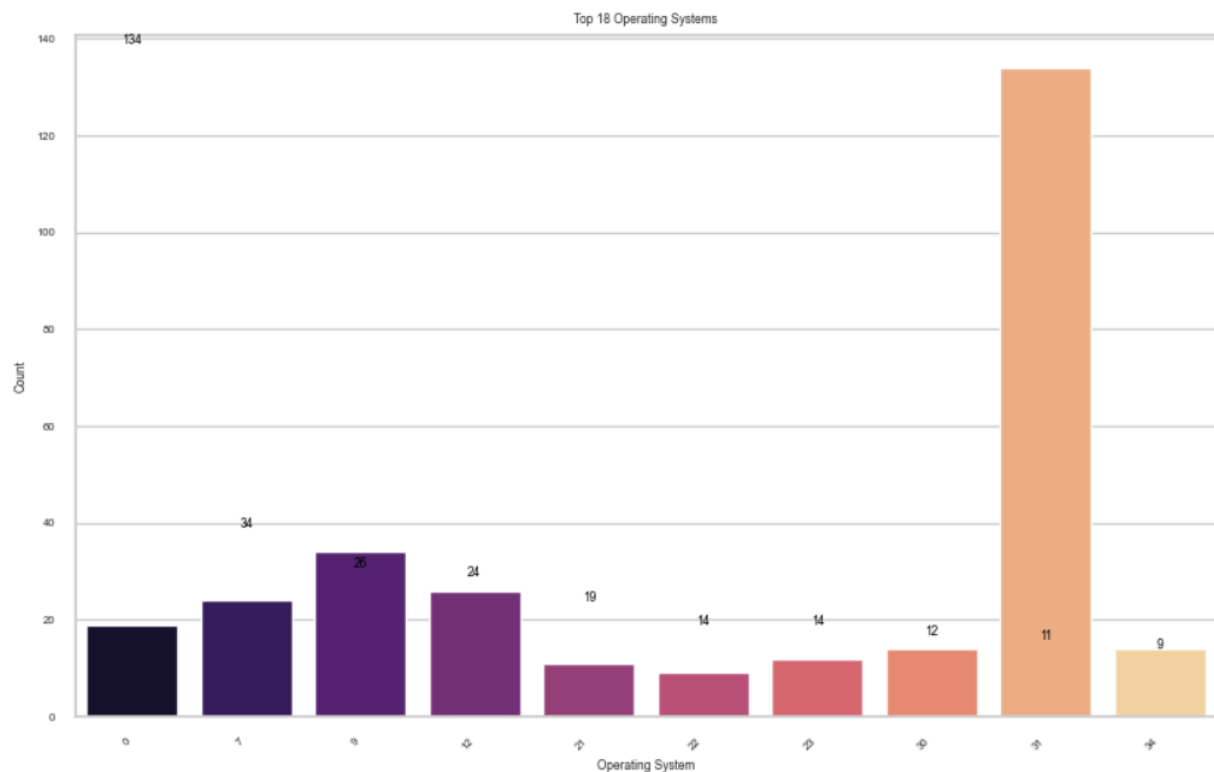


```

sns.set_style('whitegrid')
top_os=df['Operating System'].value_counts().index[:10]
os_counts = df['Operating System'].value_counts().values[:10]
fig, ax = plt.subplots (figsize=(10, 6))
ax = sns.barplot(x=top_os, y=os_counts, palette='magma')
ax.set(xlabel='Operating System', ylabel='Count', title='Top 18 Operating Systems')
plt.xticks(rotation=45, ha='right')
for i, v in enumerate(os_counts):
    ax.text(i, v+5, str(v), color='black', ha='center')
plt.show()

```

This code generates a bar plot showing the top 10 operating systems used by customers of a particular product. The data is obtained from a dataframe called 'data', and the plot is generated using the Seaborn library. The x-axis shows the name of the operating systems and the y-axis shows the count of each operating system. The plot also includes labels on the bars to indicate the exact count of each operating system. The plot is displayed using the matplotlib library



## Bivariate analysis



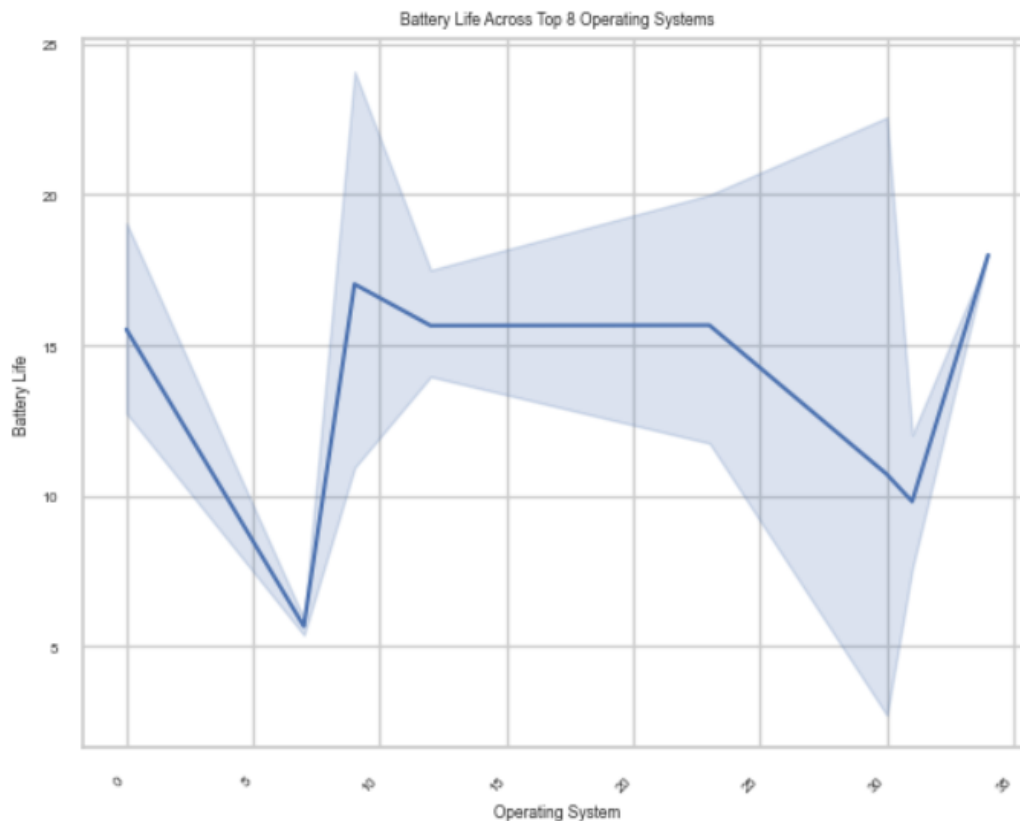
Bivariate analysis is a statistical method used to analyse the relationship between two variables in a dataset. This analysis focuses on examining how changes in one variable are related to changes in another variable

```
import seaborn as sns
import matplotlib.pyplot as plt

top_operating_systems = df['Operating System'].value_counts().head(8).index.tolist()
data_top_operating_systems = df[df['Operating System'].isin(top_operating_systems)]

sns.set_style('whitegrid')
sns.lineplot(x='Operating System', y='Battery Life', data=data_top_operating_systems)
plt.xlabel('Operating System')
plt.ylabel('Battery Life')
plt.title('Battery Life Across Top 8 Operating Systems')
plt.xticks(rotation=45, ha='right')
plt.show()
```

This code is creating a line plot to show the battery life of the top 8 operating systems. It first filters the data to include only the top 8 operating systems by frequency count. Then, it creates a line plot where the x-axis represents the operating system and the y-axis represents the battery life

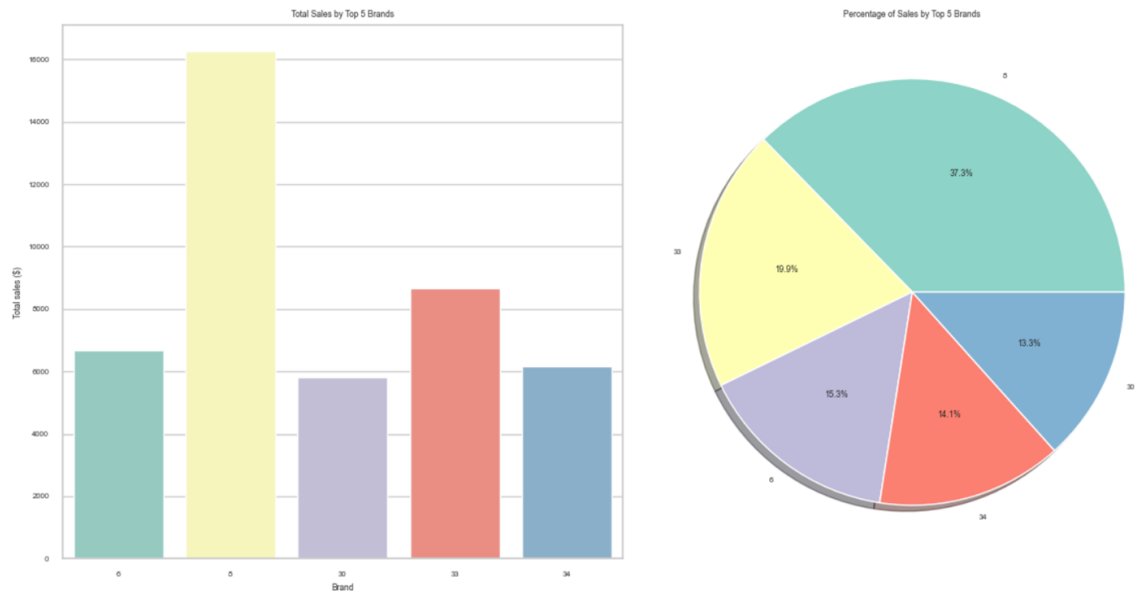


## Multivariate analysis

Multivariate analysis is a statistical technique used to analyse data that involves more than two variables. It aims to understand the relationships between multiple variables in a dataset by examining how they are related to each other and how they contribute to a particular outcome or phenomenon

```
total_sales = df.groupby('Brand')['Price'].sum().reset_index()
top_brands = total_sales.sort_values('Price', ascending=False).head(5)
top_brands['Percent'] = (top_brands['Price'] / top_brands['Price'].sum()) * 100
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))
sns.barplot(x='Brand', y='Price', data=top_brands, palette='Set3', ax=axes[0])
axes[0].set_xlabel('Brand')
axes[0].set_ylabel('Total sales ($)')
axes[0].set_title('Total Sales by Top 5 Brands')
colors = sns.color_palette('Set3', top_brands.shape[0]).as_hex()
axes[1].pie(top_brands['Percent'], labels=top_brands['Brand'], colors=colors, autopct='%1.1f%%', shadow=True)
axes[1].set_title('Percentage of Sales by Top 5 Brands')
fig.tight_layout()
plt.show()
```

This code generates a bar chart and a pie chart to display the total sales and percentage of sales for the top 5 brands in a dataset. First, the code calculates the total sales for each brand and sorts the brands by total sales in descending order. Then, it calculates the percentage of sales for each of the top 5 brands. Next, it creates a grid with two subplots, one for the bar chart and the other for the pie chart. The bar chart shows the total sales for each of the top 5 brands, while the pie chart shows the percentage of sales for each brand. Finally, it adjusts the spacing between subplots and displays the plot.



### Splitting data into train and test

First split the dataset into X and y and then split the data set. The 'X' corresponds to independent features and 'y' corresponds to the target variable.

```
X= df.drop(['Price'],axis=1)
X
```

	Brand	Model	Operating System	Connectivity	Display Type	Display Size	Resolution	Water Resistance	Battery Life	GPS	NFC
0	1	127	34	2	17	1.9	27	50.0	18.0	1	1
1	30	36	31	2	0	1.4	31	50.0	40.0	1	1
2	8	105	9	1	0	1.3	30	50.0	11.0	1	0
3	6	109	7	1	0	1.6	19	50.0	6.0	1	1
4	7	43	31	1	0	1.3	30	30.0	24.0	1	1
...	...	...	...	...	...	...	...	...	...	...	...
374	38	79	32	1	16	1.4	21	50.0	30.0	0	1
375	41	132	33	2	0	1.4	32	50.0	15.0	1	1
376	9	119	12	1	0	1.4	32	50.0	25.0	1	1
377	26	118	5	1	0	1.6	17	50.0	14.0	0	1
378	35	71	31	2	0	1.4	32	50.0	72.0	1	1

379 rows × 11 columns

For splitting training and testing data we are using `train_test_split()` function from `sklearn`. As parameters, we are passing `x`, `y`, `test_size`, `random_state`.

```
y=df['Price']
```

```
y
```

```
0      399.0
```

```
1      249.0
```

```
2      399.0
```

```
3      229.0
```

```
4      299.0
```

```
...
```

```
374    279.0
```

```
375    349.0
```

```
376    249.0
```

```
377    159.0
```

```
378    299.0
```

```
Name: Price, Length: 379, dtype: float64
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=25)
```

```
print(X_train.shape)
```

```
print(y_train.shape)
```

```
print(X_test.shape)
```

```
print(y_test.shape)
```

```
(303, 11)
```

```
(303,)
```

```
(76, 11)
```

```
(76,)
```

## 8. PERFORMANCE TESTING

### **8.1 Performance Metrics:**

Performance metrics are quantitative measures used to assess and evaluate the effectiveness, efficiency, and overall success of a system, process, or product. In the context of smartwatches or technology products in general, performance metrics provide insights into various aspects of the device's functionality, user experience, and impact.

## 1. Check model performance on test and train data for each model

To check the model performance on test and train data, we can use the "score" method to calculate the coefficient of determination (R-squared) between the predicted and actual values and also calculate the RMSE score. Comparing the R-squared scores for both the test and train data can help us determine if the model is over fitting or under fitting. The RMSE score tells us how far off the predictions of the model are, on average, from the actual values.

### 1.1 Linear Regression Model

```
# training score
```

```
error_score_lr_train= r2_score(y_train,predict_train)
print("R2 error is:",error_score_lr_train)
mse=mean_squared_error(y_train,predict_train)
rmse_lr_train=np.sqrt(mse)
print('Root Mean Squared Error:',rmse_lr_train)
```

```
R2 error is: 0.2295546632471358
```

```
Root Mean Squared Error: 179.89481395578446
```

```
# testing score
```

```
error_score_lr_test=r2_score(y_test,predict_test)
print("R2 error is:",error_score_lr_test)
mse=mean_squared_error(y_test,predict_test)
rmse_lr_test=np.sqrt(mse)
print("Root Mean Squared Error:",rmse_lr_test)
```

```
R2 error is: 0.16590308669836784
```

```
Root Mean Squared Error: 172.2507837673408
```

## 1.2 Decision Tree Regress or Model

```
# training Score
```

```
error_score_dtr_train=r2_score(y_train,predict_train_dtr)
print("R2 error is:",error_score_dtr_train)
mse=mean_squared_error(y_train,predict_train_dtr)
rmse_dtr_train=np.sqrt(mse)
print('Root Mean Squared Error:',rmse_dtr_train)
```

```
R2 error is: 0.3429614927518523
```

```
Root Mean Squared Error: 166.12811592656647
```

```
#testing score
```

```
error_score_dtr_test=r2_score(y_test,predict_test_dtr)
print("R2 error is:",error_score_dtr_test)
mse=mean_squared_error(y_test,predict_test_dtr)
rmse_dtr_test=np.sqrt(mse)
print('Root Mean Squared Error:',rmse_dtr_test)
```

```
R2 error is: 0.18085636154493812
```

```
Root Mean Squared Error: 170.69978774403583
```

## 1.3 Random Forest Regression Model

```
# training Score
```

```
error_score_rfr_train=r2_score(y_train,predict_train_rfr)
print("R2 error is:",error_score_rfr_train)
mse=mean_squared_error(y_train,predict_train_rfr)
rmse_rfr_train=np.sqrt(mse)
print('Root Mean Squared Error:',rmse_rfr_train)
```

R2 error is: 0.49758319869121215

Root Mean Squared Error: 145.2712940837865

```
#testing score
```

```
error_score_rfr_test=r2_score(y_test,predict_test_rfr)
print("R2 error is:",error_score_rfr_test)
mse=mean_squared_error(y_test,predict_test_rfr)
rmse_rfr_test=np.sqrt(mse)
print('Root Mean Squared Error:',rmse_rfr_test)
```

R2 error is: 0.4261481054255444

Root Mean Squared Error: 142.87388678752063

## 1.4 Gradient Boosting Regression Model

```
# training Score
```

```
error_score_gbr_train=r2_score(y_train,predict_train_gbr)
print("R2 error is:",error_score_gbr_train)
mse=mean_squared_error(y_train,predict_train_gbr)
rmse_gbr_train=np.sqrt(mse)
print('Root Mean Squared Error:',rmse_gbr_train)
```

R2 error is: 0.41649950162493654

Root Mean Squared Error: 156.55550514239548

```
#testing score
```

```
error_score_gbr_test=r2_score(y_test,predict_test_gbr)
print("R2 error is:",error_score_gbr_test)
mse=mean_squared_error(y_test,predict_test_gbr)
rmse_gbr_test=np.sqrt(mse)
print('Root Mean Squared Error:',rmse_gbr_test)
```

R2 error is: 0.4141301569073099

Root Mean Squared Error: 144.36220988703522

## 1.5 Extreme Gradient Boost Regression Model

```
# training Score
```

```
error_score_xgb_train=r2_score(y_train,predict_train_xgb)
print("R2 error is:",error_score_xgb_train)
mse=mean_squared_error(y_train,predict_train_xgb)
rmse_xgb_train=np.sqrt(mse)
print('Root Mean Squared Error:',rmse_xgb_train)
```

R2 error is: 0.9219250785587862

Root Mean Squared Error: 57.26687782255483

```
#testing score
```

```
error_score_xgb_test=r2_score(y_test,predict_test_xgb)
print("R2 error is:",error_score_xgb_test)
mse=mean_squared_error(y_test,predict_test_xgb)
rmse_xgb_test=np.sqrt(mse)
print('Root Mean Squared Error:',rmse_xgb_test)
```

R2 error is: 0.805145250048114

Root Mean Squared Error: 83.2546401826171

## 2. Comparing models



```

results=pd.DataFrame(columns=['Model','Training R2','Testing R2','Traing RMSE','Testing RMSE'])
results.loc[0]=['Linear Regression',error_score_lr_train,error_score_lr_test,rmse_lr_train,rmse_lr_test]
results.loc[1]=['DecisionTreeRegressor',error_score_dtr_train,error_score_dtr_test,rmse_dtr_train,rmse_dtr_test]
results.loc[2]=['RandomForestRegressor',error_score_rfr_train,error_score_rfr_test,rmse_rfr_train,rmse_rfr_test]
results.loc[3]=['GradientBoostingRegressor',error_score_gbr_train,error_score_gbr_test,rmse_gbr_train,rmse_gbr_test]
results.loc[4]=['XG Boost Regressor',error_score_xgb_train,error_score_xgb_test,rmse_xgb_train,rmse_xgb_test]
print(results)

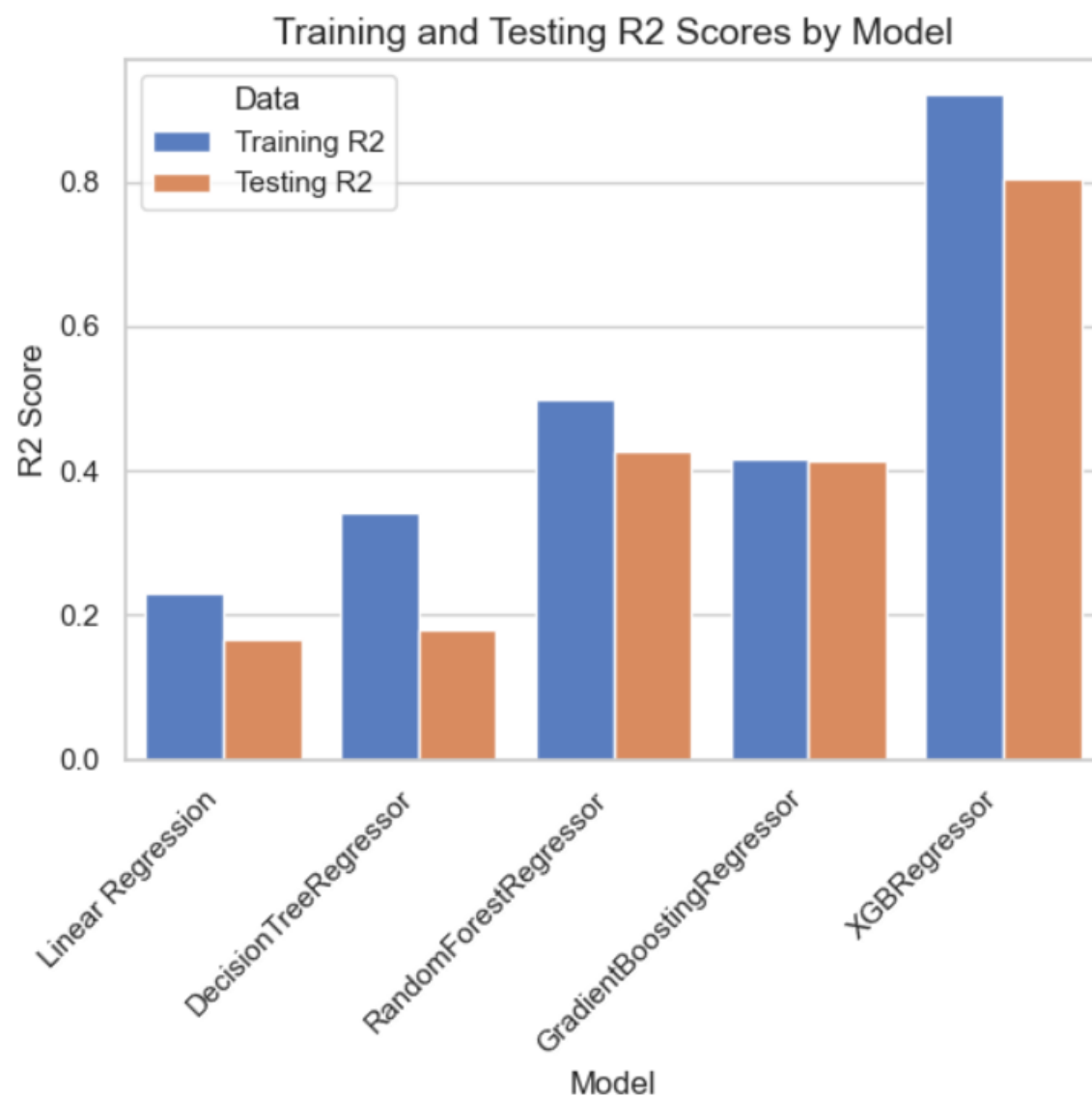
```

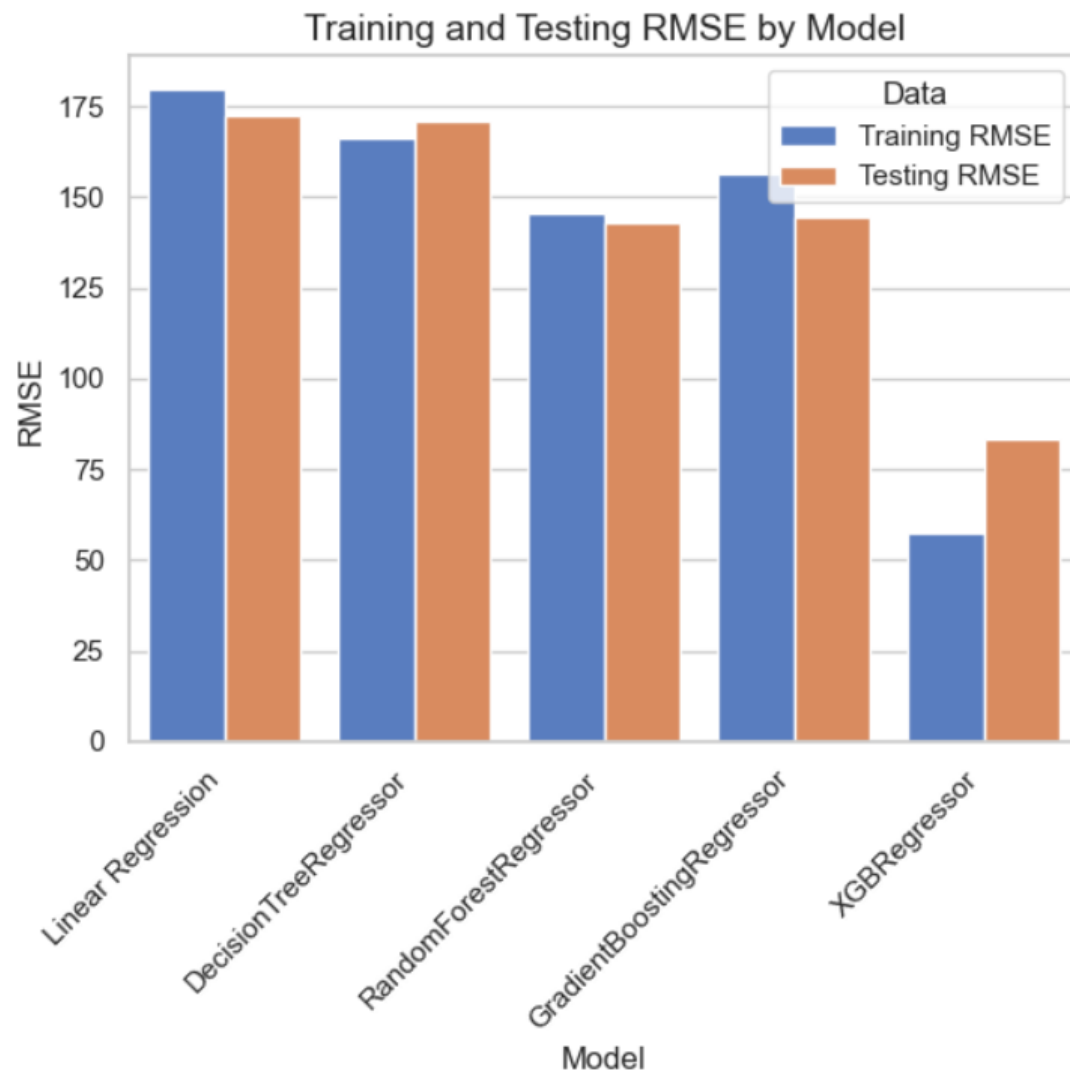
	Model	Training R2	Testing R2	Traing RMSE \
0	Linear Regression	0.229555	0.165903	179.894814
1	DecisionTreeRegressor	0.342961	0.180856	166.128116
2	RandomForestRegressor	0.497583	0.426148	145.271294
3	GradientBoostingRegressor	0.416500	0.414130	156.555505
4	XG Boost Regressor	0.921925	0.805145	57.266878

	Testing RMSE
0	172.250784
1	170.699788
2	142.873887
3	144.362210
4	83.254640

This code creates a Pandas Data Frame named "results" that contains the model names, R-squared scores, and root mean squared errors (RMSE) for both the training and testing data for each of the five regression models: Linear Regression, Decision Tree Regression, Random Forest Regressor, Gradient Boosting Regressor, and XG Boost Regressor.





## **9. RESULTS**

### **9.1 Output Screenshot:**

1: Save the best model

```
# dumping the selection model  
pickle.dump(xgb,open('SW.pkl','wb'))
```

This code uses the "pickle" library in Python to save the trained XG Boost Regressor model named "xgb" as a file named "SW.pkl".

## 2: Integrate with Web Framework

we will be building a web application that would help us integrate the machine learning model we have built and trained. A user interface is provided for the users to enter the values for predictions. The entered values are fed into the saved model, and the prediction is displayed on the UI.

Building HTML pages

Building server side script

Run the web application

### Activity 2.1: Building Html Pages:

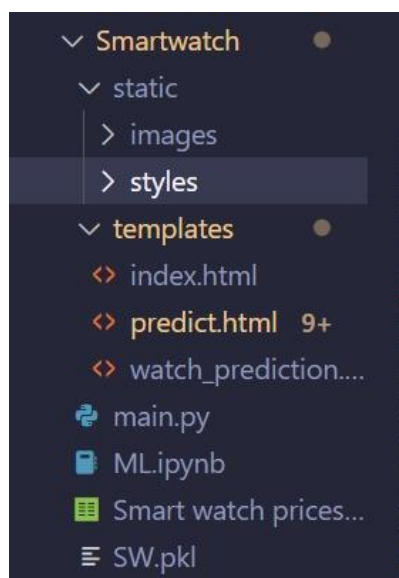
For this project we create three html files:

index.html

predict.html

watch\_prediction.html

and save these html files in the templates folder



## 2.2: Build Python code:

```
import pickle
from flask import Flask, render_template, request
import pandas as pd
import numpy as np
```

```
model1= pickle.load(open('SW.pkl','rb'))
app= Flask(__name__)
@app.route('/')
def home():
    return render_template('index.html')
```

```
@app.route('/predict')
def index():
    return render_template('predict.html')
```

```
@app.route('/data_predict',methods=['GET','POST'])
def predict():
    if request.method == 'POST':
        Brand=request.form['Brand']

        if Brand == 'Garmin':
            Brand = 8
        if Brand == 'Mobvoi':
            Brand = 18
        if Brand == 'Fitbit':
            Brand = 6
        if Brand == 'Fossil':
            Brand = 7
        if Brand == 'Amazfit':
            Brand = 0
        if Brand == 'Samsung':
            Brand = 30
        if Brand == 'Huawei':
            Brand = 10
        if Brand == 'TicWatch':
            Brand = 35
        if Brand == 'Xiaomi':
            Brand = 36
        if Brand == 'Skagen':
            Brand = 31
        if Brand == 'Suunto':
            Brand = 33
        if Brand == 'Honor':
            Brand = 9
        if Brand == 'Apple':
            Brand = 1
        if Brand == 'Polar':
            Brand = 27
        if Brand == 'Casio':
            Brand = 3
```

```
if Brand == 'Withings':  
    Brand = 38  
if Brand == 'Oppo':  
    Brand = 26  
if Brand == 'Timex':  
    Brand = 37  
if Brand == 'Diesel':  
    Brand = 4  
if Brand == 'Misfit':  
    Brand = 17  
if Brand == 'Michael Kors':  
    Brand = 16  
if Brand == 'Zepp':  
    Brand = 41  
if Brand == 'LG':  
    Brand = 13  
if Brand == 'TAG Heuer':  
    Brand = 34  
if Brand == 'Asus':  
    Brand = 2  
if Brand == 'Montblanc':  
    Brand = 19  
if Brand == 'Sony':  
    Brand = 32  
if Brand == 'Realme':  
    Brand = 29  
if Brand == 'Matrix':  
    Brand = 15  
if Brand == 'Kate Spade':  
    Brand = 11  
if Brand == 'Kospet':  
    Brand = 12
```

```
if model == 'Hybrid HR':  
    model = 44  
if model == 'Venu Sq':  
    model = 106  
if model == 'MagicWatch 2':  
    model = 56  
if model == 'Ticwatch Pro 3':  
    model = 97  
if model == 'Vapor X':  
    model = 104  
if model == 'Z':  
    model = 132  
  
os = request.form['Operating System']  
if os == 'Wear OS':  
    os = 31  
if os == 'Garmin OS':  
    os = 9  
if os == 'Lite OS':  
    os = 12
```

```

connect = request.form['Connectivity']
if connect == 'Bluetooth, Wi-Fi':
    connect = 1
if connect == 'Bluetooth, Wi-Fi, Cellular':
    connect = 2
if connect == 'Bluetooth':
    connect = 0
if connect == 'Bluetooth, Wi-Fi, GPS':
    connect = 3
if connect == 'Bluetooth, Wi-Fi, NFC':
    connect = 4

display_type = request.form['Display Type']
if display_type == 'AMOLED':
    display_type = 0
if display_type == 'LCD':
    display_type = 9

```

```

prediction = model1.predict(pd.DataFrame([[brand,model,os,connect,display_type,display_size,resolution,water,battery,gps,nfc]],
                                         columns= ['Brand', 'Model', 'Operating System', 'Connectivity',
                                                    'Display Type', 'Display Size', 'Resolution',
                                                    'Water Resistance', 'Battery Life', 'GPS', 'NFC'])))

prediction = np.round(prediction,2)

return render_template('watch_prediction.html', prediction_text = "is {}".format(prediction))

```

Main Function:

This code sets the entry point of the Flask application. The function "app.run()" is called, which starts the Flask development server.

```

if __name__ == '__main__':
    app.run()

```

## 2.3: Run the web application

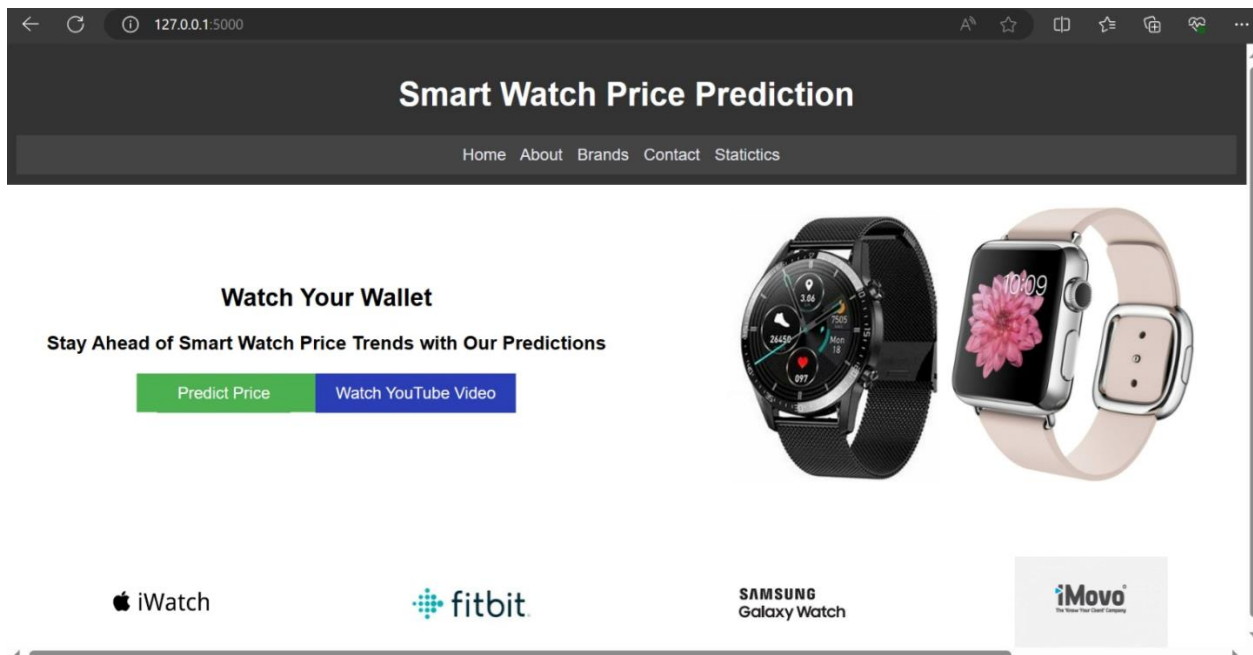
When you run the "main.py" file this window will open in the output terminal. Copy the <http://127.0.0.1:5000> and paste this link in your browser.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

File "c:\Users\Pallavi\OneDrive\Desktop\Smartwatch\main.py", line 6, in <module>
    model1= pickle.load(open('SW.pkl','rb'))
FileNotFoundError: [Errno 2] No such file or directory: 'SW.pkl'
PS C:\Users\Pallavi\OneDrive\Desktop> cd C:\Users\Pallavi\OneDrive\Desktop\Smartwatch
PS C:\Users\Pallavi\OneDrive\Desktop\Smartwatch> python main.py
* Serving Flask app 'main'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 313-869-208
127.0.0.1 - - [18/Nov/2023 17:44:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2023 17:44:26] "GET /static/images/img1.jpg HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2023 17:44:26] "GET /static/styles/style.css HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2023 17:44:26] "GET /static/images/fitbiting.jfif HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2023 17:44:26] "GET /static/images/samsung.jfif HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2023 17:44:26] "GET /static/images/apple-iwatch.jpg HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2023 17:44:26] "GET /static/images/iMOMO-600x367.png HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2023 17:44:26] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [18/Nov/2023 17:44:28] "GET /predict.html HTTP/1.1" 404 -
127.0.0.1 - - [18/Nov/2023 17:44:33] "GET /predict HTTP/1.1" 200 -
Input DataFrame: [558.6463]
Predicted Price: [558.65]
127.0.0.1 - - [18/Nov/2023 17:45:00] "POST /data_predict HTTP/1.1" 200 -
```

This is the "index.html" file that appears when we paste the URL into the browser. To proceed to the next page, click the 'Predict Price' button.





Smart Watch Price Prediction Home

### Fill the features of Smart Watch

Watch Brand:  
TicWatch

Watch Model:  
Access Runway

Operating System:  
Android Wear

Connectivity:  
Bluetooth, Wi-Fi, Cellular

Display Type:  
PMOLED

Display Size:  
1.36

Resolution:  
320 x 320

## Smart Watch Price Prediction

 Home

Connectivity:  
Bluetooth, Wi-Fi, Cellular

Display Type:  
PMOLED

Display Size:  
1.36

Resolution:  
320 x 320

Water Resistance:  
100

Battery Life:  
3

GPS:  
Yes

NFC:  
Yes

Submit

These are all the features we have given to to predict the smart watch price.

## Smart Watch Price Prediction

 Home [Input Data](#)

### Predicted Smartwatch Price:

Predicted Smartwatch Price in \$ is [435.95]

#### Smart Watch Price Prediction

**Designers:** Pallavi, Indhu, Varshini, Divya

**Email:** mandadipallavi@gmail.com

**University:** Vellore Institute of Technology

**Address:** Amaravathi, Andhra Pradesh

#### Useful Links

- [Home Page](#)
- [User data page](#)
- [Amazon Purchase Link](#)

#### Social Media

Connect with us :     

So finally this is our predicted smartwatch price in \$ is 435.95 , according to the features we have given.

## **10. ADVANTAGES & DISADVANTAGES**

Advantages:

Innovation and Technological Advancements:

Forecasting the future allows for anticipation of technological innovations, enabling

Market Insight:

Accurate forecasting provides valuable insights into market trends, enabling manufacturers to align their products with consumer preferences and demands.

Competitive Positioning:

Companies can strategically position themselves in the market by forecasting and adapting to expected changes in smartwatch features, functionality, and pricing.

Competitive Positioning:

Forecasting helps set realistic consumer expectations regarding upcoming smartwatch models, fostering transparency and trust in the market.

Disadvantages:

Rapid Technological Changes:

The fast-paced nature of technology can make accurate forecasting challenging, as new developments may emerge unexpectedly, affecting product features and prices.

Uncertain Market Factors:

External factors such as economic conditions, global events, and unexpected competition can introduce uncertainty, making it difficult to accurately predict market dynamics.

Dependency on Supply Chain:

Disruptions in the supply chain, such as component shortages or geopolitical issues, can impact production costs and, subsequently, smartwatch prices.

#### Consumer Behavior Changes:

Changes in consumer preferences or unforeseen shifts in behavior can lead to unexpected demand patterns, affecting the accuracy of price forecasts.

#### Market Saturation:

If the market becomes saturated with similar products, accurately forecasting consumer adoption rates becomes challenging, potentially leading to overproduction.

### **11. Conclusion**

Smartwatches in Horology 2.0 are likely to strengthen their integration within larger technological ecosystems. Seamless connectivity with smartphones, smart home devices, and other gadgets could add value, potentially influencing pricing strategies based on the level of ecosystem integration. Within the domain of Horology 2.0, predicting the course of smartwatch prices becomes an essential navigational aid in an environment that is changing quickly. The conclusion of this study shows how market forces, consumer expectations, and technological innovation will dynamically interact to shape the pricing paradigm of the future. The study points out trends that emphasize the importance of personalized experiences and health-centric features, but it also recognizes the difficulties and unknowns involved in projecting the pricing trajectory. However, it emphasizes how important these projections are for stakeholders, stressing how successful navigation of the ever-changing smartwatch market depends on flexible tactics, ongoing innovation, and a deep awareness of changing consumer preferences.

### **12. FUTURE SCOPE**

There is a great deal of room for further research and development in the future application of Horology 2.0 for predicting smartwatch prices. Further research into the incorporation of cutting-edge technologies like artificial intelligence (AI), augmented reality, and sustainable materials into smartwatches may be necessary as consumer preferences change and technology continues to progress quickly. Furthermore, a compelling research avenue is to examine how pricing strategies

are affected by geopolitical shifts, regulatory changes, and global economic trends. Furthermore, improving data analytics, predictive algorithms, and real-time market monitoring can help forecasting models be refined even further, improving the precision and dependability of smartwatch price predictions and providing industry stakeholders with important information in a constantly shifting market. Continued improvements in processing power, battery life, and display technology may lead to more sophisticated and capable smartwatches, potentially affecting pricing based on the level of innovation. Smartwatches adopting modular or upgradeable designs, allowing users to replace or upgrade certain components, may impact pricing models, giving users more flexibility in choosing features.

### **13. APPENDIX**

An appendix in a document typically contains supplementary information, additional details, or supporting materials that are relevant to the main content but not essential for understanding it.

#### **Survey Data:**

In support of our forecast for the future of smartwatch prices, we conducted a user survey to gather insights into consumer preferences and expectations. The survey focused on key factors influencing smartwatch purchasing decisions, including design preferences, desired features, and perceived value.

#### **Industry Trends:**

A compilation of recent industry reports and analyses from reputable sources, such as market research firms and technology publications, is provided to offer additional context to the forecast. These reports highlight emerging technologies, market trends, and competitive landscapes shaping the smartwatch industry.

### source code:

```
import pickle

from flask import Flask, render_template, request

import pandas as pd

import numpy as np

model1= pickle.load(open('SW.pkl','rb'))

app= Flask(__name__)

@app.route('/')

def home():

    return render_template('index.html')

@app.route('/predict')

def index():

    return render_template('predict.html')

@app.route('/data_predict',methods=['GET','POST'])

def predict():

    if request.method == 'POST':

        Brand=request.form['Brand']

        if Brand == 'Garmin':

            Brand = 8

        if Brand == 'Mobvoi':

            Brand = 18

        if Brand == 'Fitbit':

            Brand = 6

        if Brand == 'Fossil':

            Brand = 7

        if Brand == 'Amazfit':

            Brand = 0

        if Brand == 'Samsung':

            Brand = 30

        if Brand == 'Huawei':

            Brand = 10

        if Brand == 'TicWatch':

            Brand = 35

        if Brand == 'Xiaomi':

            Brand = 36

        if Brand == 'Skagen':
```

```
Brand = 31
if Brand == 'Suunto':
    Brand = 33
if Brand == 'Honor':
    Brand = 9
if Brand == 'Apple':
    Brand = 1
if Brand == 'Polar':
    Brand = 27
if Brand == 'Casio':
    Brand = 3
if Brand == 'Withings':
    Brand = 38
if Brand == 'Oppo':
    Brand = 26
if Brand == 'Timex':
    Brand = 37
if Brand == 'Diesel':
    Brand = 4
if Brand == 'Misfit':
    Brand = 17
if Brand == 'Michael Kors':
    Brand = 16
if Brand == 'Zepp':
    Brand = 41
if Brand == 'LG':
    Brand = 13
if Brand == 'TAG Heuer':
    Brand = 34
if Brand == 'Asus':
    Brand = 2
if Brand == 'Montblanc':
    Brand = 19
if Brand == 'Sony':
    Brand = 32
if Brand == 'Realme':
    Brand = 29
```

```
if Brand == 'Matrix':  
    Brand = 15  
if Brand == 'Kate Spade':  
    Brand = 11  
if Brand == 'Kospet':  
    Brand = 12  
if Brand == 'Emporio Armani':  
    Brand = 5  
if Brand == 'Nokia':  
    Brand = 23  
if Brand == 'Ticwatch':  
    Brand = 36  
if Brand == 'MyKronoz':  
    Brand = 22  
if Brand == 'Zeblaze':  
    Brand = 40  
if Brand == 'Lemfo':  
    Brand = 14  
if Brand == 'Nubia':  
    Brand = 24  
if Brand == 'Moto':  
    Brand = 20  
if Brand == 'Polaroid':  
    Brand = 28  
if Brand == 'Motorola':  
    Brand = 21  
if Brand == 'OnePlus':  
    Brand = 25
```

```
Model =request.form['Model']
```

```
if Model == '7':
```

```
    Model = 0
```

```
if Model == '9 Baro':
```

```
    Model = 1
```

```
if Model == '9 Peak':
```

```
    Model = 2
```

```
if Model == 'Access Bradshaw 2':
```



```
Model = 3
if Model == 'Access Gen 5':
    Model = 4
if Model == 'Access Runway':
    Model = 5
if Model == 'Alpha':
    Model = 6
if Model == 'Bip S':
    Model = 7
if Model == 'Bip U Pro':
    Model = 8
if Model == 'C2':
    Model = 9
if Model == 'C2+':
    Model = 10
if Model == 'Charge 5':
    Model = 11
if Model == 'Collider':
    Model = 12
if Model == 'Collider HR':
    Model = 13
if Model == 'Connected':
    Model = 14
if Model == 'Connected Modular 45':
    Model = 15
if Model == 'Cosmo':
    Model = 16
if Model == 'E2':
    Model = 17
if Model == 'E3':
    Model = 18
if Model == 'Enduro':
    Model = 19
if Model == 'Fadelite':
    Model = 20
if Model == 'Falster 2':
    Model = 21
```

```
if Model == 'Falster 3':  
    Model = 22  
  
if Model == 'Fenix 6 Pro Solar':  
    Model = 23  
  
if Model == 'Forerunner 945':  
    Model = 24  
  
if Model == 'Forerunner 945 LTE':  
    Model = 25  
  
if Model == 'G-Shock GBD-H1000':  
    Model = 26  
  
if Model == 'G-Shock GSW-H1000':  
    Model = 27  
  
if Model == 'G-Shock Move':  
    Model = 28  
  
if Model == 'GTR 2':  
    Model = 29  
  
if Model == 'GTR 2e':  
    Model = 30  
  
if Model == 'GTR 3':  
    Model = 31  
  
if Model == 'GTS 2':  
    Model = 32  
  
if Model == 'GTS 2e':  
    Model = 33  
  
if Model == 'Galaxy Watch':  
    Model = 34  
  
if Model == 'Galaxy Watch 3':  
    Model = 35  
  
if Model == 'Galaxy Watch 4':  
    Model = 36  
  
if Model == 'Galaxy Watch Active 2':  
    Model = 37  
  
if Model == 'Galaxy Watch Active2':  
    Model = 38  
  
if Model == 'Galaxy Watch3':  
    Model = 39  
  
if Model == 'Gear S3 Frontier':
```

```
Model = 40
if Model == 'Gen 5':
    Model = 41
if Model == 'Gen 5 Carlyle':
    Model = 42
if Model == 'Gen 6':
    Model = 43
if Model == 'Hybrid HR':
    Model = 44
if Model == 'Ignite':
    Model = 45
if Model == 'Ignite 2':
    Model = 46
if Model == 'Instinct':
    Model = 47
if Model == 'Instinct Solar':
    Model = 48
if Model == 'Ionic':
    Model = 49
if Model == 'Ironman R300 GPS':
    Model = 50
if Model == 'Jorn Hybrid HR':
    Model = 51
if Model == 'Lem12 Pro':
    Model = 52
if Model == 'Lexington 2':
    Model = 53
if Model == 'Lily':
    Model = 54
if Model == 'Magic Watch 2':
    Model = 55
if Model == 'MagicWatch 2':
    Model = 56
if Model == 'Metropolitan R':
    Model = 57
if Model == 'Metropolitan RAMOLED':
    Model = 58
```

```
if Model == 'Metropolitan+':  
    Model = 59  
  
if Model == 'Mi Watch':  
    Model = 60  
  
if Model == 'Mi Watch Lite':  
    Model = 61  
  
if Model == 'Mi Watch Revolve':  
    Model = 62  
  
if Model == 'Moto 360':  
    Model = 63  
  
if Model == 'Moto 360 (3rd Gen)':  
    Model = 64  
  
if Model == 'On Axial':  
    Model = 65  
  
if Model == 'On Fadelite':  
    Model = 66  
  
if Model == 'On Full Guard 2.5':  
    Model = 67  
  
if Model == 'Optimus Pro':  
    Model = 68  
  
if Model == 'PowerWatch 2':  
    Model = 69  
  
if Model == 'Prime 2':  
    Model = 70  
  
if Model == 'Pro 3':  
    Model = 71  
  
if Model == 'Pro 3 GPS':  
    Model = 72  
  
if Model == 'Pro 4G/LTE':  
    Model = 73  
  
if Model == 'Pro Trek F30':  
    Model = 74  
  
if Model == 'Pro Trek Smart WSD-30F':  
    Model = 75  
  
if Model == 'Pro Trek Smart WSD-F30':  
    Model = 76  
  
if Model == 'Pro Trek WSD-F30':
```

```
Model = 77
if Model == 'Scallop 2':
    Model = 78
if Model == 'ScanWatch':
    Model = 79
if Model == 'ScanWatch 38mm':
    Model = 80
if Model == 'Sense':
    Model = 81
if Model == 'SmartWatch 3':
    Model = 82
if Model == 'Smartwatch 3':
    Model = 83
if Model == 'Sport':
    Model = 84
if Model == 'Steel HR':
    Model = 85
if Model == 'Steel HR Sport':
    Model = 86
if Model == 'Stratos 3':
    Model = 87
if Model == 'Summit 2':
    Model = 88
if Model == 'Summit 2+':
    Model = 89
if Model == 'T-Rex Pro':
    Model = 90
if Model == 'Thor 5 Pro':
    Model = 91
if Model == 'TicWatch C2+':
    Model = 92
if Model == 'TicWatch E2':
    Model = 93
if Model == 'TicWatch E3':
    Model = 94
if Model == 'TicWatch Pro':
    Model = 95
```

if Model == 'TicWatch Pro 2020':

Model = 96

if Model == 'TicWatch Pro 3':

Model = 97

if Model == 'TicWatch Pro 3 GPS':

Model = 98

if Model == 'TicWatch S2':

Model = 99

if Model == 'Vantage M':

Model = 100

if Model == 'Vantage M2':

Model = 101

if Model == 'Vantage V2':

Model = 102

if Model == 'Vapor 2':

Model = 103

if Model == 'Vapor X':

Model = 104

if Model == 'Venu 2':

Model = 105

if Model == 'Venu Sq':

Model = 106

if Model == 'Venu Sq Music':

Model = 107

if Model == 'Versa 2':

Model = 108

if Model == 'Versa 3':

Model = 109

if Model == 'Versa Lite Edition':

Model = 110

if Model == 'Vivoactive 4':

Model = 111

if Model == 'Watch':

Model = 112

if Model == 'Watch 2':

Model = 113

if Model == 'Watch 2 ECG Edition':

```
Model = 114
if Model == 'Watch 3 Pro':
    Model = 115
if Model == 'Watch ES':
    Model = 116
if Model == 'Watch Fit':
    Model = 117
if Model == 'Watch Free':
    Model = 118
if Model == 'Watch GS Pro':
    Model = 119
if Model == 'Watch GT 2':
    Model = 120
if Model == 'Watch GT 2 Pro':
    Model = 121
if Model == 'Watch GT 2e':
    Model = 122
if Model == 'Watch GT2 Pro':
    Model = 123
if Model == 'Watch S Pro':
    Model = 124
if Model == 'Watch SE':
    Model = 125
if Model == 'Watch Series 6':
    Model = 126
if Model == 'Watch Series 7':
    Model = 127
if Model == 'Watch Urbane 2':
    Model = 128
if Model == 'Watch Urbane 2nd Ed.':
    Model = 129
if Model == 'Watch Urbane 2nd Edition':
    Model = 130
if Model == 'Watch W7':
    Model = 131
if Model == 'Z':
    Model = 132
```

```
if Model == 'Z Titanium':  
    Model = 133  
  
if Model == 'ZeTime':  
    Model = 134  
  
if Model == 'ZeTime Elite 2':  
    Model = 135  
  
if Model == 'ZenWatch 3':  
    Model = 136  
  
Operating_System = request.form['Operating_System']  
  
if Operating_System == 'Amazfit OS':  
    Operating_System = 0  
  
if Operating_System == 'Android':  
    Operating_System = 1  
  
if Operating_System == 'Android OS':  
    Operating_System = 2  
  
if Operating_System == 'Android Wear':  
    Operating_System = 3  
  
if Operating_System == 'Casio OS':  
    Operating_System = 4  
  
if Operating_System == 'ColorOS':  
    Operating_System = 5  
  
if Operating_System == 'Custom OS':  
    Operating_System = 6  
  
if Operating_System == 'Fitbit OS':  
    Operating_System = 7  
  
if Operating_System == 'Fossil OS':  
    Operating_System = 8  
  
if Operating_System == 'Garmin OS':  
    Operating_System = 9  
  
if Operating_System == 'HarmonyOS':  
    Operating_System = 10  
  
if Operating_System == 'Hybrid OS':  
    Operating_System = 11  
  
if Operating_System == 'Lite OS':  
    Operating_System = 12  
  
if Operating_System == 'LiteOS':  
    Operating_System = 13
```



```
if Operating_System == 'MIUI':  
    Operating_System = 14  
if Operating_System == 'MIUI For Watch':  
    Operating_System = 15  
if Operating_System == 'MIUI for Watch':  
    Operating_System = 16  
if Operating_System == 'Matrix OS':  
    Operating_System = 17  
if Operating_System == 'Mi Wear OS':  
    Operating_System = 18  
if Operating_System == 'MyKronoz OS':  
    Operating_System = 19  
if Operating_System == 'Nubia OS':  
    Operating_System = 20  
if Operating_System == 'Polar OS':  
    Operating_System = 21  
if Operating_System == 'Proprietary':  
    Operating_System = 22  
if Operating_System == 'Proprietary OS':  
    Operating_System = 23  
if Operating_System == 'RTOS':  
    Operating_System = 24  
if Operating_System == 'Realme OS':  
    Operating_System = 25  
if Operating_System == 'Skagen OS':  
    Operating_System = 26  
if Operating_System == 'Suunto OS':  
    Operating_System = 27  
if Operating_System == 'Timex OS':  
    Operating_System = 28  
if Operating_System == 'Tizen':  
    Operating_System = 29  
if Operating_System == 'Tizen OS':  
    Operating_System = 30  
if Operating_System == 'Wear OS':  
    Operating_System = 31  
if Operating_System == 'Withings OS':
```

```
Operating_System = 32
if Operating_System == 'Zepp OS':
    Operating_System = 33
if Operating_System == 'watchOS':
    Operating_System = 34
Connectivity = request.form['Connectivity']
if Connectivity == 'Bluetooth, Wi-Fi':
    Connectivity = 1
if Connectivity == 'Bluetooth, Wi-Fi, Cellular':
    Connectivity = 2
if Connectivity == 'Bluetooth':
    Connectivity = 0
if Connectivity == 'Bluetooth, Wi-Fi, GPS':
    Connectivity = 3
if Connectivity == 'Bluetooth,Wi-Fi, NFC':
    Connectivity = 4
Display_Type = request.form['Display_Type']
if Display_Type == 'AMOLED':
    Display_Type = 0
if Display_Type == 'Analog':
    Display_Type = 1
if Display_Type == 'Color Touch':
    Display_Type = 2
if Display_Type == 'Dual Layer':
    Display_Type = 3
if Display_Type == 'E-Ink':
    Display_Type = 4
if Display_Type == 'E-ink':
    Display_Type = 5
if Display_Type == 'Gorilla Glass':
    Display_Type = 6
if Display_Type == 'IPS':
    Display_Type = 7
if Display_Type == 'IPS LCD':
    Display_Type = 8
if Display_Type == 'LCD':
    Display_Type = 9
```

```
if Display_Type == 'MIP':
    Display_Type = 10
if Display_Type == 'Memory LCD':
    Display_Type = 11
if Display_Type == 'Memory-in-pixel (MIP)':
    Display_Type = 12
if Display_Type == 'Monochrome':
    Display_Type = 13
if Display_Type == 'OLED':
    Display_Type = 14
if Display_Type == 'P-OLED':
    Display_Type = 15
if Display_Type == 'PMOLED':
    Display_Type = 16
if Display_Type == 'Retina':
    Display_Type = 17
if Display_Type == 'STN LCD':
    Display_Type = 18
if Display_Type == 'Sunlight-visible':
    Display_Type = 19
if Display_Type == 'Sunlight-visible, transflective memory-in-pixel (MIP)':
    Display_Type = 20
if Display_Type == 'Super AMOLED':
    Display_Type = 21
if Display_Type == 'TFT':
    Display_Type = 22
if Display_Type == 'TFT LCD':
    Display_Type = 23
if Display_Type == 'TFT-LCD':
    Display_Type = 24
if Display_Type == 'Transflective':
    Display_Type = 25
if Display_Type == 'transflective':
    Display_Type = 26
```

```
Display_Size = float(request.form['Display_Size'])
```

```
if Display_Size == 0.9:
    Display_Size = 0
if Display_Size == 1.0:
    Display_Size = 1
if Display_Size == 1.04:
    Display_Size = 2
if Display_Size == 1.06:
    Display_Size = 3
if Display_Size == 1.1:
    Display_Size = 4
if Display_Size == 1.19:
    Display_Size = 5
if Display_Size == 1.2:
    Display_Size = 6
if Display_Size == 1.22:
    Display_Size = 7
if Display_Size == 1.23:
    Display_Size = 8
if Display_Size == 1.28:
    Display_Size = 9
if Display_Size == 1.3:
    Display_Size = 10
if Display_Size == 1.32:
    Display_Size = 11
if Display_Size == 1.34:
    Display_Size = 12
if Display_Size == 1.35:
    Display_Size = 13
if Display_Size == 1.36:
    Display_Size = 14
if Display_Size == 1.363164893617021:
    Display_Size = 15
if Display_Size == 1.38:
    Display_Size = 16
if Display_Size == 1.39:
    Display_Size = 17
if Display_Size == 1.4:
```

```
    Display_Size = 18
if Display_Size == 1.42:
    Display_Size = 19
if Display_Size == 1.43:
    Display_Size = 20
if Display_Size == 1.5:
    Display_Size = 21
if Display_Size == 1.57:
    Display_Size = 22
if Display_Size == 1.58:
    Display_Size = 23
if Display_Size == 1.6:
    Display_Size = 24
if Display_Size == 1.64:
    Display_Size = 25
if Display_Size == 1.65:
    Display_Size = 26
if Display_Size == 1.75:
    Display_Size = 27
if Display_Size == 1.78:
    Display_Size = 28
if Display_Size == 1.9:
    Display_Size = 29
if Display_Size == 1.91:
    Display_Size = 30
if Display_Size == 2.1:
    Display_Size = 31
if Display_Size == 4.01:
    Display_Size = 32

Resolution = request.form['Resolution']

if Resolution == '126 x 36':
    Resolution = 0
if Resolution == '128 x 128':
    Resolution = 1
if Resolution == '160 x 160':
```

```
Resolution = 2
if Resolution == '176 x 176':
    Resolution = 3
if Resolution == '200 x 200':
    Resolution = 4
if Resolution == '228 x 172':
    Resolution = 5
if Resolution == '240 x 198':
    Resolution = 6
if Resolution == '240 x 201':
    Resolution = 7
if Resolution == '240 x 240':
    Resolution = 8
if Resolution == '260 x 260':
    Resolution = 9
if Resolution == '280 x 280':
    Resolution = 10
if Resolution == '280 x 456':
    Resolution = 11
if Resolution == '300 x 300':
    Resolution = 12
if Resolution == '320 x 300':
    Resolution = 13
if Resolution == '320 x 302':
    Resolution = 14
if Resolution == '320 x 320':
    Resolution = 15
if Resolution == '324 x 394':
    Resolution = 16
if Resolution == '326 x 326':
    Resolution = 17
if Resolution == '328 x 328':
    Resolution = 18
if Resolution == '336 x 336':
    Resolution = 19
if Resolution == '348 x 250':
    Resolution = 20
```

```
if Resolution == '348 x 442':
```

```
    Resolution = 21
```

```
if Resolution == '360 x 360':
```

```
    Resolution = 22
```

```
if Resolution == '368 x 448':
```

```
    Resolution = 23
```

```
if Resolution == '372 x 430':
```

```
    Resolution = 24
```

```
if Resolution == '390 x 390':
```

```
    Resolution = 25
```

```
if Resolution == '394 x 324':
```

```
    Resolution = 26
```

```
if Resolution == '396 x 484':
```

```
    Resolution = 27
```

```
if Resolution == '400 x 400':
```

```
    Resolution = 28
```

```
if Resolution == '402 x 476':
```

```
    Resolution = 29
```

```
if Resolution == '416 x 416':
```

```
    Resolution = 30
```

```
if Resolution == '450 x 450':
```

```
    Resolution = 31
```

```
if Resolution == '454 x 454':
```

```
    Resolution = 32
```

```
if Resolution == '466 x 466':
```

```
    Resolution = 33
```

```
if Resolution == '480 x 480':
```

```
    Resolution = 34
```

```
if Resolution == '960 x 192':
```

```
    Resolution = 35
```

```
Water_Resistance = int(request.form['Water_Resistance'])
```

```
if Water_Resistance == 50:
```

```
    Water_Resistance = 5
```

```
if Water_Resistance == 30:
```

```
    Water_Resistance = 4
```

```
if Water_Resistance == 100:
```

```
    Water_Resistance = 2
```

```
if Water_Resistance == 200:
```

```
    Water_Resistance = 3
```

```
if Water_Resistance == 15:
```

```
    Water_Resistance = 0
```

```
if Water_Resistance == 'Not Specified':
```

```
    Water_Resistance = 6
```

```
if Water_Resistance == 10:
```

```
    Water_Resistance = 1
```

```
Battery_Life = request.form['Battery_Life']
```

```
if Battery_Life == '1':
```

```
    Battery_Life = 0
```

```
if Battery_Life == '1.5':
```

```
    Battery_Life = 1
```

```
if Battery_Life == '10':
```

```
    Battery_Life = 2
```

```
if Battery_Life == '11':
```

```
    Battery_Life = 3
```

```
if Battery_Life == '12':
```

```
    Battery_Life = 4
```

```
if Battery_Life == '14':
```

```
    Battery_Life = 5
```

```
if Battery_Life == '15':
```

```
    Battery_Life = 6
```

```
if Battery_Life == '16':
```

```
    Battery_Life = 7
```

```
if Battery_Life == '18':
```

```
    Battery_Life = 8
```

```
if Battery_Life == '2':
```

```
    Battery_Life = 9
```

```
if Battery_Life == '20':
```

```
    Battery_Life = 10
```

```
if Battery_Life == '24':
```

```
    Battery_Life = 11
```



```
if Battery_Life == '25':  
    Battery_Life = 12  
if Battery_Life == '3':  
    Battery_Life = 13  
if Battery_Life == '30':  
    Battery_Life = 14  
if Battery_Life == '4':  
    Battery_Life = 15  
if Battery_Life == '40':  
    Battery_Life = 16  
if Battery_Life == '45':  
    Battery_Life = 17  
if Battery_Life == '48':  
    Battery_Life = 18  
if Battery_Life == '48 hours':  
    Battery_Life = 19  
if Battery_Life == '5':  
    Battery_Life = 20  
if Battery_Life == '56':  
    Battery_Life = 21  
if Battery_Life == '6':  
    Battery_Life = 22  
if Battery_Life == '60':  
    Battery_Life = 23  
if Battery_Life == '7':  
    Battery_Life = 24  
if Battery_Life == '70':  
    Battery_Life = 25  
if Battery_Life == '72':  
    Battery_Life = 26  
if Battery_Life == '8':  
    Battery_Life = 27  
if Battery_Life == '9':  
    Battery_Life = 28  
if Battery_Life == 'Unlimited':  
    Battery_Life = 29
```

```

GPS = request.form['GPS']

if GPS == 'Yes':

    GPS = 1

if GPS == 'No':

    GPS = 0

NFC = request.form['NFC']

if NFC == 'Yes':

    NFC = 1

if NFC == 'No':

    NFC = 0

predict_price = model1.predict(pd.DataFrame([[Brand, Model, Operating_System, Connectivity, Display_Type, Display_Size, Resolution,
Water_Resistance, Battery_Life, GPS, NFC]],

        columns=['Brand', 'Model', 'Operating_System', 'Connectivity', 'Display_Type', 'Display_Size', 'Resolution',
'Water_Resistance', 'Battery_Life', 'GPS', 'NFC']))

print("Input DataFrame:", predict_price)

predict_price = np.round(predict_price,2)

print("Predicted Price:", predict_price)

return render_template('watch_prediction.html', prediction_text="Predicted Smartwatch Price in $ is {}".format(predict_price))

else:

    return render_template('predict.html')

if __name__ == '__main__':

    app.run(debug=True)

```

## **GitHub & Demo Link :**

### **GitHub Link**

<https://github.com/smartinternz02/Sl-GuidedProject-612118-1698753989>

### **Demo Link**

<https://drive.google.com/file/d/1fXVJUeoJjQdwQLATJiRjwBFix2SyPbo8/view?usp=sharing>