

**FDS [20CD101] COURSE PROJECT REPORT**

**On**

***“Food Delivery Time Prediction”***

Developed By:

H.T.NO STUDENT NAME

2203A54014 M. SUHAS

2203A54004 K. BRUHADWI RAO

2203A54023 V. GANESH

2203A54027 M. NAGA SRUJAN

2203A54029 M. VASAVI REDDY

Under the Guidance of

Mrs.Asiya, M.Tech.(Ph.D)

Assistant Professor

Submitted to

Department Computer Science and Artificial Intelligence SR University

Ananthasagar(V), Hasanparthy(M), Hanamkonda(Dist.) – 506371 [www.sru.edu.in](http://www.sru.edu.in/)

**November 2023**

**Department of Computer Science and Artificial Intelligence**

**CERTIFICATE**

This is to certify that the FDS course project report entitled **“Food Delivery Time Prediction”** is a record of bonafide work carried out by the student(s) M.Suhas, K.Bruhadwi Rao, V.Ganesh, M.Naga Srujan & M.Vasavi reddy bearing roll number(s) 2203A54014, 2203A54004, 2203A54023, 2203A54027 & 2203A54029 of Computer Science and Artificial Intelligence department during the academic year 2022-23.

**Supervisor**

(Asiya)

**INDEX**

**Sl. No Title Page No.**

1. Problem statement 1
2. Existing System 2
3. Disadvantage of Existing system 3
4. Proposed system 5
5. Modules 7
6. Knowledge required 8
7. Source code (.ipyb file code) 9
8. Results 12
9. Conclusion 16
10. References 17

# PROBLEM STATEMENT

The accurate prediction of food delivery time is a crucial aspect for food delivery services like Zomato and Swiggy to maintain transparency and customer satisfaction. These companies employ Machine Learning algorithms to forecast delivery times by leveraging historical data on the time taken by delivery partners for similar distances. However, there remains a need to enhance the accuracy and reliability of these predictions. The problem at hand is to develop a more robust and efficient Machine Learning model that can precisely estimate food delivery times in real-time by considering factors such as distance between food preparation and delivery locations and historical delivery time data. This project aims to address this challenge by providing insights and guidance on using Python for food delivery time prediction through Machine Learning techniques.

# Existing System

# The existing system of food delivery time prediction, as depicted in the provided code, is a basic implementation that uses a Long Short-Term Memory (LSTM) neural network to estimate the time it will take for food to be delivered to customers. The key components of the existing system include:

# Data Input

# The system relies on historical data, which includes information about the delivery person's age, ratings, geographic coordinates (latitude and longitude), and past delivery times.

# Geospatial Calculation

# The system calculates the distance between the restaurant and the delivery location using the Haversine formula, which is based on latitude and longitude.

# Machine Learning Model

# It employs a simple LSTM neural network for time prediction, which takes into account the delivery person's age, ratings, and distance as input features.

# Data Visualization:

# The system visualizes the relationships between various input factors and delivery times using scatter plots and statistical analysis.

# Training and Prediction:

# The model is trained on the historical data, and users can input specific values for age, ratings, and distance to receive a predicted delivery time.

# The system, while functional, has limitations related to accuracy, adaptability, and user experience, as discussed in previous responses. To improve the system's performance and user satisfaction, various enhancements and advanced techniques can be implemented.

# Disadvantage of existing System

# The existing system for food delivery time prediction, as described in the provided code, has several disadvantages and limitations:

# Simplicity of the Model

# The existing system uses a relatively simple deep learning model (LSTM) for time prediction. This simplicity may not capture the complexity of real-world factors that affect food delivery times.

# Data Dependency

# The accuracy of the predictions heavily relies on the availability and quality of historical data. Inconsistent or limited historical data can lead to unreliable predictions.

# Inability to Adapt to Real-Time Changes

# The system does not adapt well to real-time changes in traffic conditions, weather, road closures, or other unforeseen events that can significantly impact delivery times.

# Lack of Geospatial Features

# While the code calculates distances based on latitude and longitude, it does not account for factors like traffic patterns, road quality, or delivery route efficiency, which can be critical in determining delivery times.

# Limited Feature Engineering

# The code uses a small set of features for prediction, which may not capture all the relevant factors affecting delivery time, such as restaurant preparation time, order volume, and demand fluctuations.

# Model Complexity

# Deep learning models like LSTM are relatively complex and might not be necessary for this specific prediction task. Simpler models might be more interpretable and sufficient for the purpose.

# Scalability Challenges

# The code does not address issues related to scaling the system to handle a larger volume of delivery data or more delivery partners.

# Interpretability

# Deep learning models like LSTM can be difficult to interpret, making it challenging to understand how the model makes its predictions.

# Lack of User Interface

# The code focuses on the modeling aspect but does not include a user interface or user experience (UX) design, which is essential for customer and delivery partner interactions.

# Data Privacy and Security

# The code does not address data privacy and security concerns, which are crucial when handling customer information and location data.

# Maintenance and Updates

# Keeping the system up-to-date with the latest data and model improvements is not considered in the code. In a real-world application, this would be a continuous and resource-intensive process.

# Failure to Address Regulatory Compliance

# The code does not account for potential regulatory requirements and compliance related to food delivery and data handling.

# To improve the existing system and make it more accurate, adaptable, and user-friendly, it would be essential to address these disadvantages by incorporating more advanced data science techniques, enhancing data collection, ensuring real-time adaptability, and considering factors such as scalability, user experience, and security.

**PROPOSED SYSTEM**

A proposed system for food delivery time prediction should address the disadvantages of the existing system and offer improvements in accuracy, scalability, user experience, and adaptability. Here's an outline of a potential system:

1. Improved Data Collection and Integration

- Collect data from various sources, including order histories, real-time traffic and weather data, and restaurant preparation times.

- Integrate geographic data to calculate accurate distances and routes.

1. Enhanced Data Quality

- Implement data quality checks and cleansing procedures to ensure accurate and reliable data for modeling.

- Handle missing or erroneous data points appropriately

1. Advanced Machine Learning Models

- Utilize advanced machine learning techniques, such as ensemble models, gradient boosting, or deep learning, to create more accurate and adaptable prediction models.

- Incorporate techniques for time series analysis to capture temporal patterns.

1. Real-time Data Integration

- Develop a system that continuously updates and integrates real-time data, such as traffic conditions, weather, and delivery progress, to make dynamic predictions.

1. Scalability

- Design the system to handle a high volume of delivery data, making it scalable to accommodate increased demand.

1. Transparency and Explanation

- Implement interpretability tools to provide explanations for the predictions made by the model, enhancing trust and transparency.

1. Customer Feedback and Review System

- Integrate a feedback system that allows customers to rate their delivery experience and provide comments, which can be used to further improve predictions and service quality.

1. Adaptive Models

- Implement models that adapt to changes in delivery patterns, traffic, and other external factors by retraining the model regularly.

1. Monitoring and Maintenance

- Set up monitoring systems to detect issues, anomalies, and model degradation, enabling proactive maintenance and updates.

1. Regulatory Compliance

- Ensure compliance with all relevant food delivery and data handling regulations in the regions where the service operates.

1. Dynamic Routing and Scheduling

- Develop algorithms for dynamic routing and scheduling that consider factors like traffic, road closures, and delivery partner

availability to optimize delivery times.

1. Delivery Partner Optimization

- Utilize data and machine learning to optimize the allocation of delivery partners to orders, taking into account factors like distance, rating, and current location.

1. Machine Learning Operations (MLOps)

- Implement MLOps practices for model deployment, monitoring, and management, making it easier to maintain and update the system.

1. Feedback Loop

- Establish a feedback loop for continuous improvement, where user feedback, delivery data, and model performance are analyzed and used to make ongoing enhancements.

The proposed system aims to provide a more accurate, efficient, and user-friendly food delivery time prediction service, addressing the shortcomings of the existing system while maintaining a strong focus on data quality, security, and regulatory compliance.

# MODULES

1. Pandas (pd)

- Purpose: Pandas is a widely used data manipulation and analysis library in Python. It provides data structures and functions to work with structured data, such as tables or spreadsheets.

- Usage in Code: In the code, Pandas is used for tasks like reading a CSV file, data exploration (with `data.head()` and `data.info()`), and creating dataframes to manage and process data.

1. NumPy (np)

- Purpose: NumPy is a fundamental library for numerical operations in Python. It provides support for arrays and mathematical functions.

- Usage in Code: NumPy is used for mathematical operations, such as converting degrees to radians and calculating distances using the Haversine formula.

1. Plotly Express (px)

- Purpose: Plotly Express is a data visualization library for creating interactive and visually appealing plots and charts.

- Usage in Code: Plotly Express is used to generate various types of plots, including scatter plots and box

plots, to visualize relationships between different variables, such as distance, time taken, age, ratings, and more.

1. scikit-learn (sklearn)

- Purpose: Scikit-learn is a machine learning library that provides a wide range of tools for tasks such as data splitting, model training, model evaluation, and feature selection.

- Usage in Code: The code imports `train\_test\_split` from scikit-learn to split the dataset into training and testing sets, which is a common step in machine learning model development.

1. Keras

- Purpose: Keras is a high-level deep learning framework that simplifies the process of building, training, and deploying neural network models.

- Usage in Code: The code uses Keras to create and train a Long Short-Term Memory (LSTM) neural network model for food delivery time prediction. It defines the model architecture, compiles it, and fits the model to the training data.

**KNOWLEDGE REQUIRED TO DEVELOP THIS APPLICATION**

* Python Programming
* Data Analysis and Manipulation
* Data Visualization
* Geospatial Analysis
* Machine Learning
* Deep Learning
* Neural Networks
* Data Collection and Integration
* Feature Engineering
* Model Evaluation

**SOURCE CODE [.ipyb FILE]:**

import pandas as pd

import numpy as np

import plotly.express as px

data = pd.read\_csv("/content/deliverytime.txt")

print(data.head())

data.info()

data.isnull().sum()

# Set the earth's radius (in kilometers)

R = 6371

# Convert degrees to radians

def deg\_to\_rad(degrees):

return degrees \* (np.pi/180)

# Function to calculate the distance between two points using the haversine formula

def distcalculate(lat1, lon1, lat2, lon2):

d\_lat = deg\_to\_rad(lat2-lat1)

d\_lon = deg\_to\_rad(lon2-lon1)

a = np.sin(d\_lat/2)\*\*2 + np.cos(deg\_to\_rad(lat1)) \* np.cos(deg\_to\_rad(lat2)) \* np.sin(d\_lon/2)\*\*2

c = 2 \* np.arctan2(np.sqrt(a), np.sqrt(1-a))

return R \* c

# Calculate the distance between each pair of points

data['distance'] = np.nan

for i in range(len(data)):

data.loc[i, 'distance'] = distcalculate(data.loc[i, 'Restaurant\_latitude'],

data.loc[i, 'Restaurant\_longitude'],

data.loc[i, 'Delivery\_location\_latitude'],

data.loc[i, 'Delivery\_location\_longitude'])

print(data.head())

figure = px.scatter(data\_frame = data,

x="distance",

y="Time\_taken(min)",

size="Time\_taken(min)",

trendline="ols",

title = "Relationship Between Distance and Time Taken")

figure.show()

figure = px.scatter(data\_frame = data,

x="Delivery\_person\_Age",

y="Time\_taken(min)",

size="Time\_taken(min)",

color = "distance",

trendline="ols",

title = "Relationship Between Time Taken and Age")

figure.show()

figure = px.scatter(data\_frame = data,

x="Delivery\_person\_Ratings",

y="Time\_taken(min)",

size="Time\_taken(min)",

color = "distance",

trendline="ols",

title = "Relationship Between Time Taken and Ratings")

figure.show()

fig = px.box(data,

x="Type\_of\_vehicle",

y="Time\_taken(min)",

color="Type\_of\_order")

fig.show()

#splitting data

from sklearn.model\_selection import train\_test\_split

x = np.array(data[["Delivery\_person\_Age",

"Delivery\_person\_Ratings",

"distance"]])

y = np.array(data[["Time\_taken(min)"]])

xtrain, xtest, ytrain, ytest = train\_test\_split(x, y,

test\_size=0.10,

random\_state=42)

# creating the LSTM neural network model

from keras.models import Sequential

from keras.layers import Dense, LSTM

model = Sequential()

model.add(LSTM(128, return\_sequences=True, input\_shape= (xtrain.shape[1], 1)))

model.add(LSTM(64, return\_sequences=False))

model.add(Dense(25))

model.add(Dense(1))

model.summary()

# training the model

model.compile(optimizer='adam', loss='mean\_squared\_error')

model.fit(xtrain, ytrain, batch\_size=1, epochs=9)

print("Food Delivery Time Prediction")

a = int(input("Age of Delivery Partner: "))

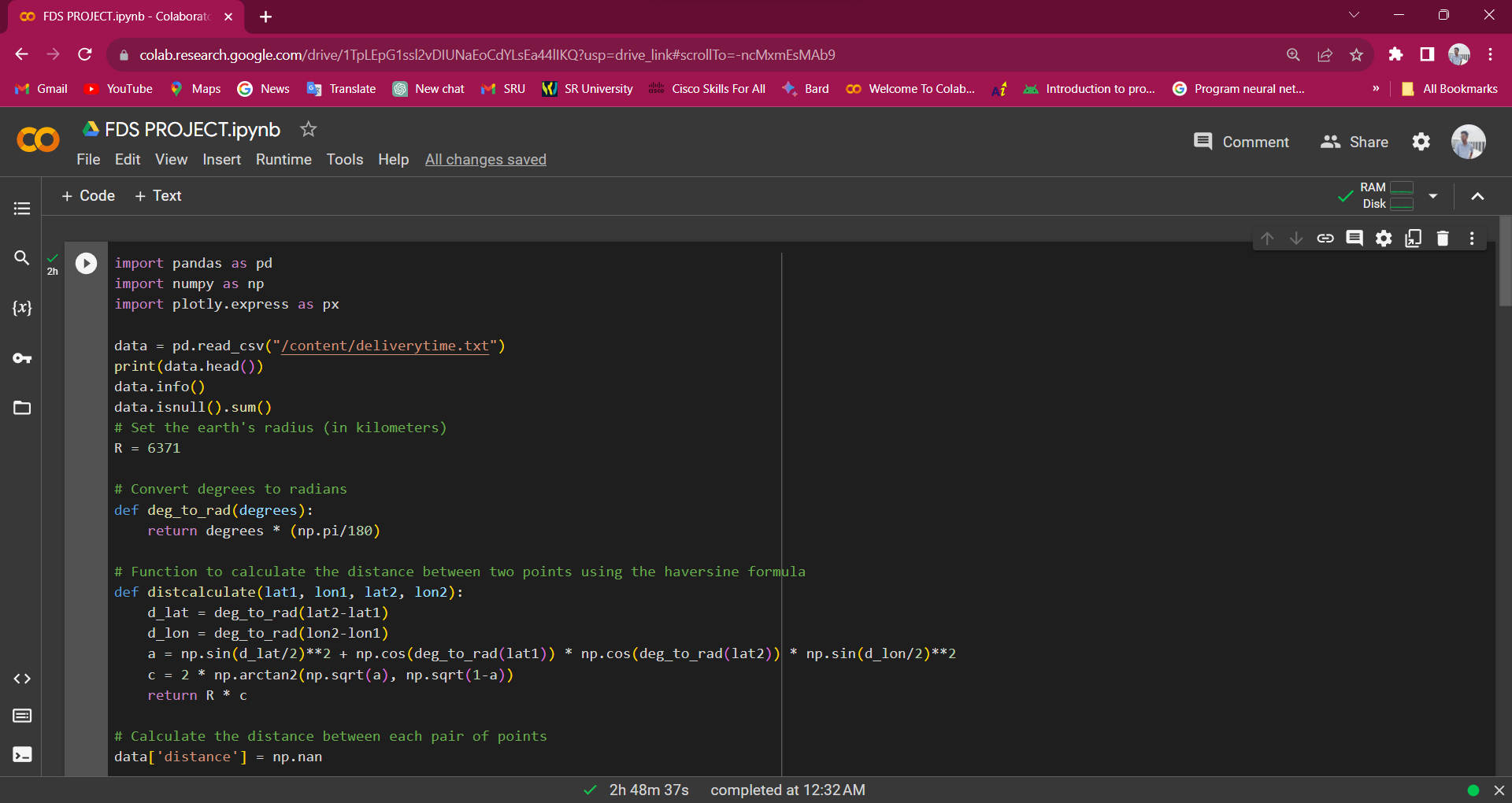
b = float(input("Ratings of Previous Deliveries: "))

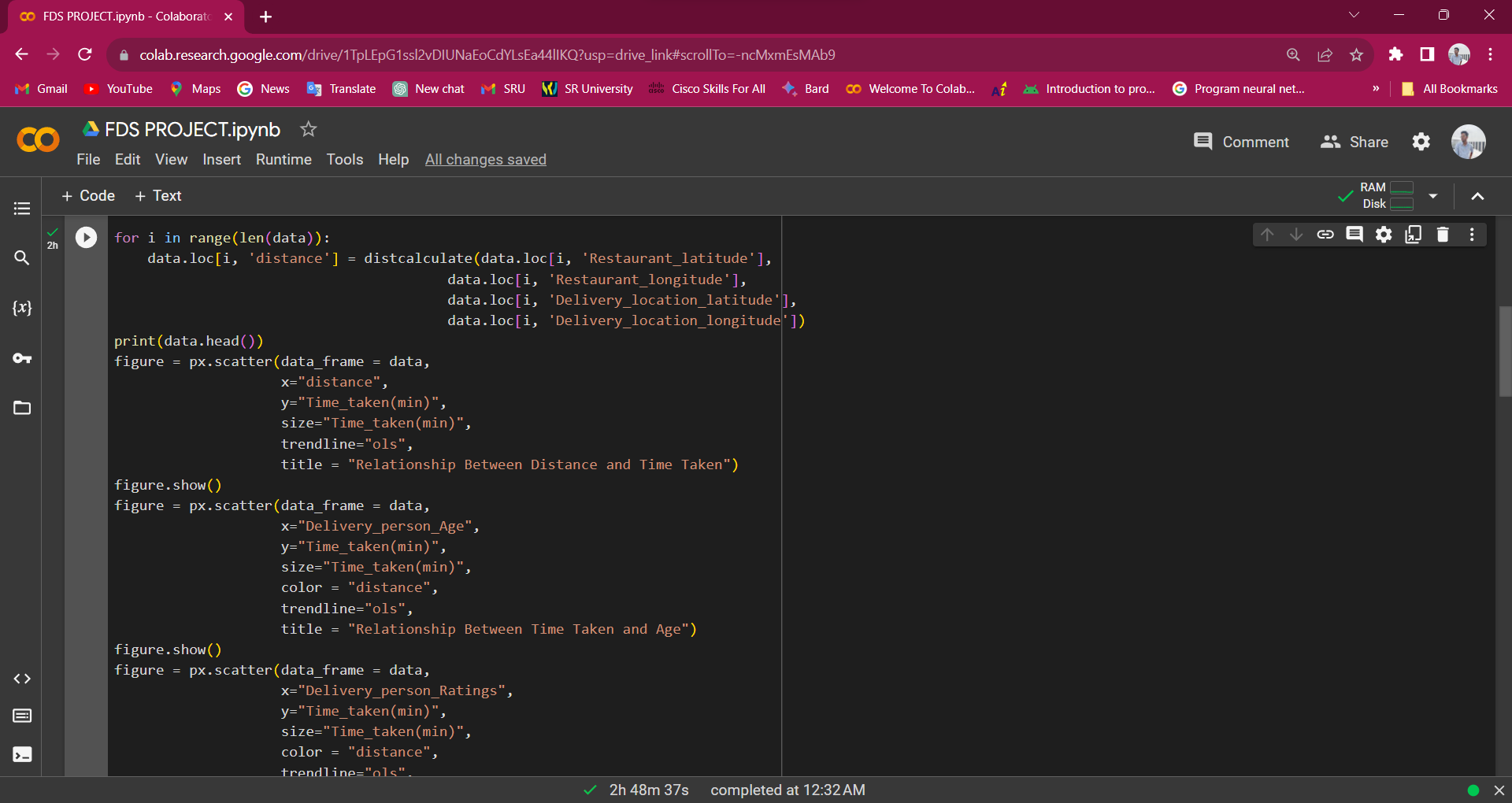
c = int(input("Total Distance: "))

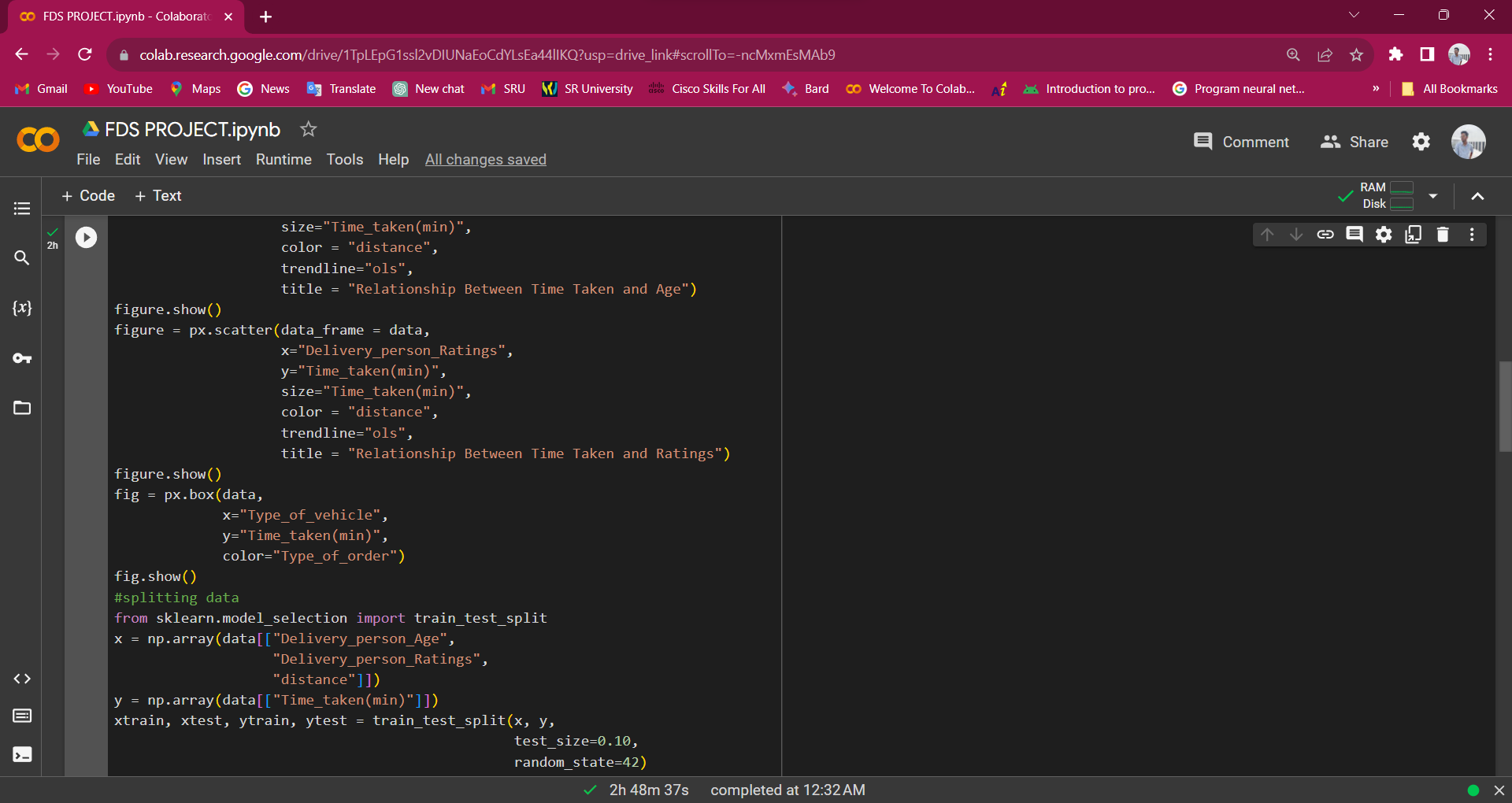
features = np.array([[a, b, c]])

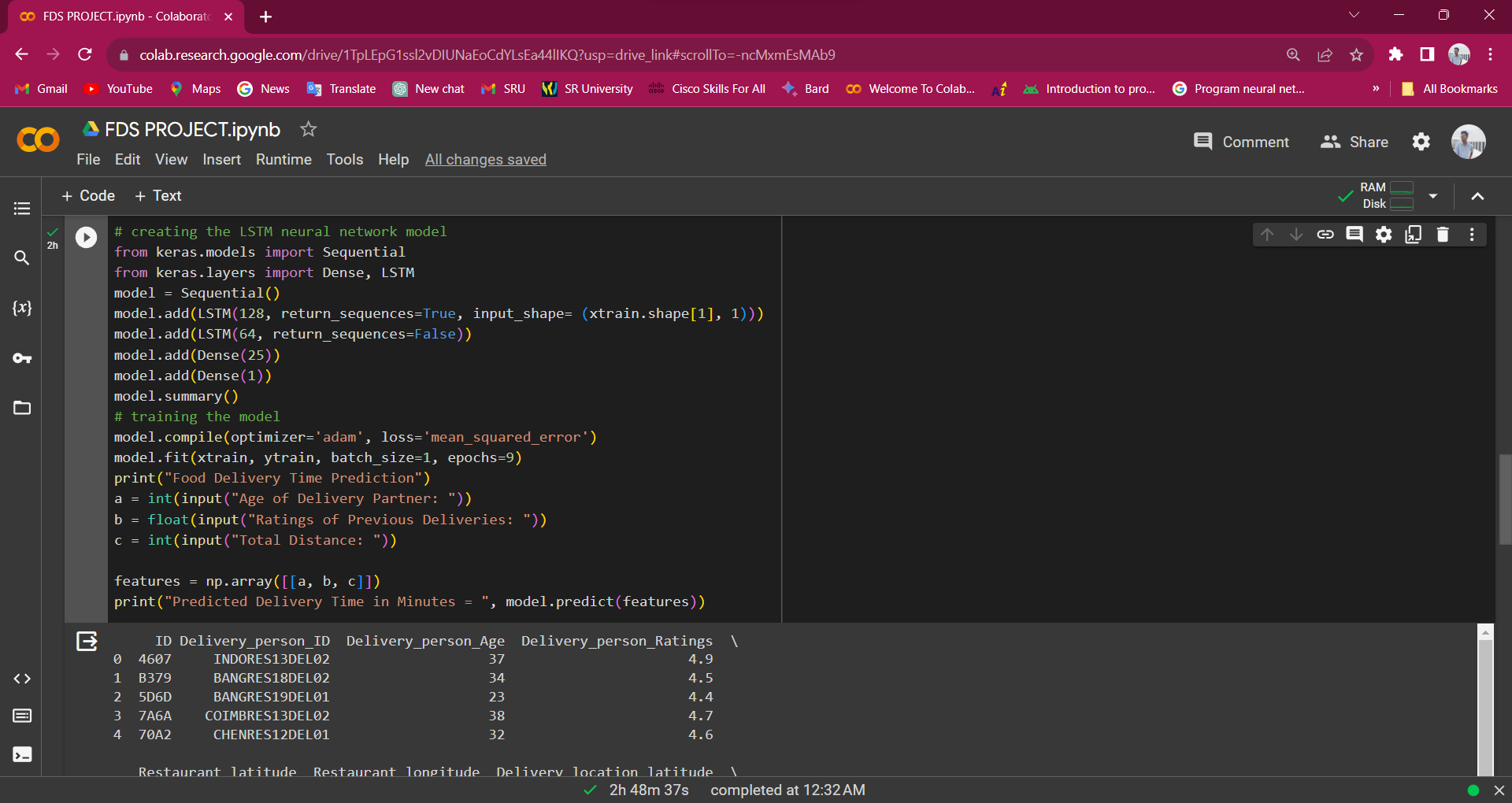
print("Predicted Delivery Time in Minutes = ", model.predict(features))

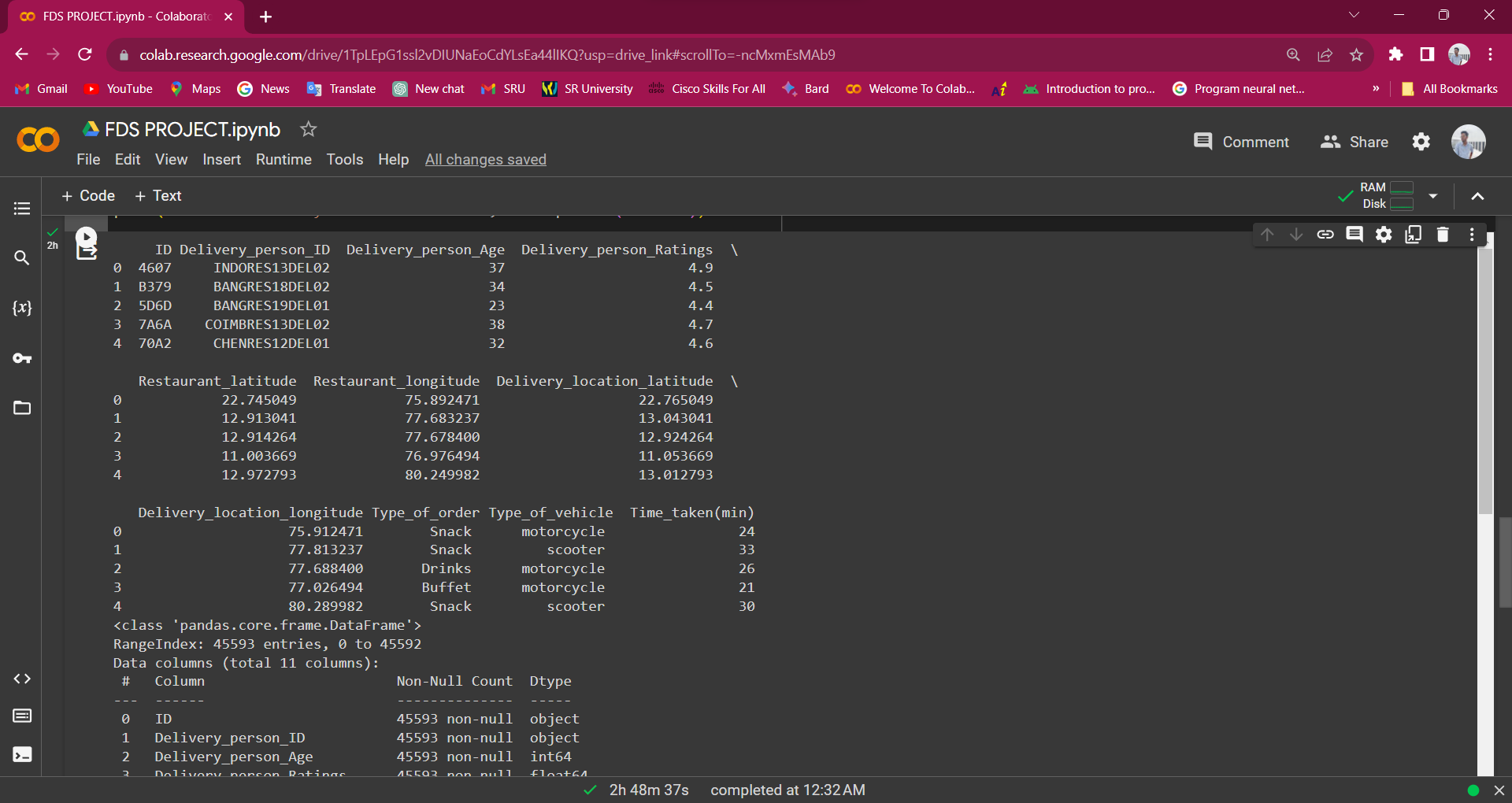
**RESULTS:**

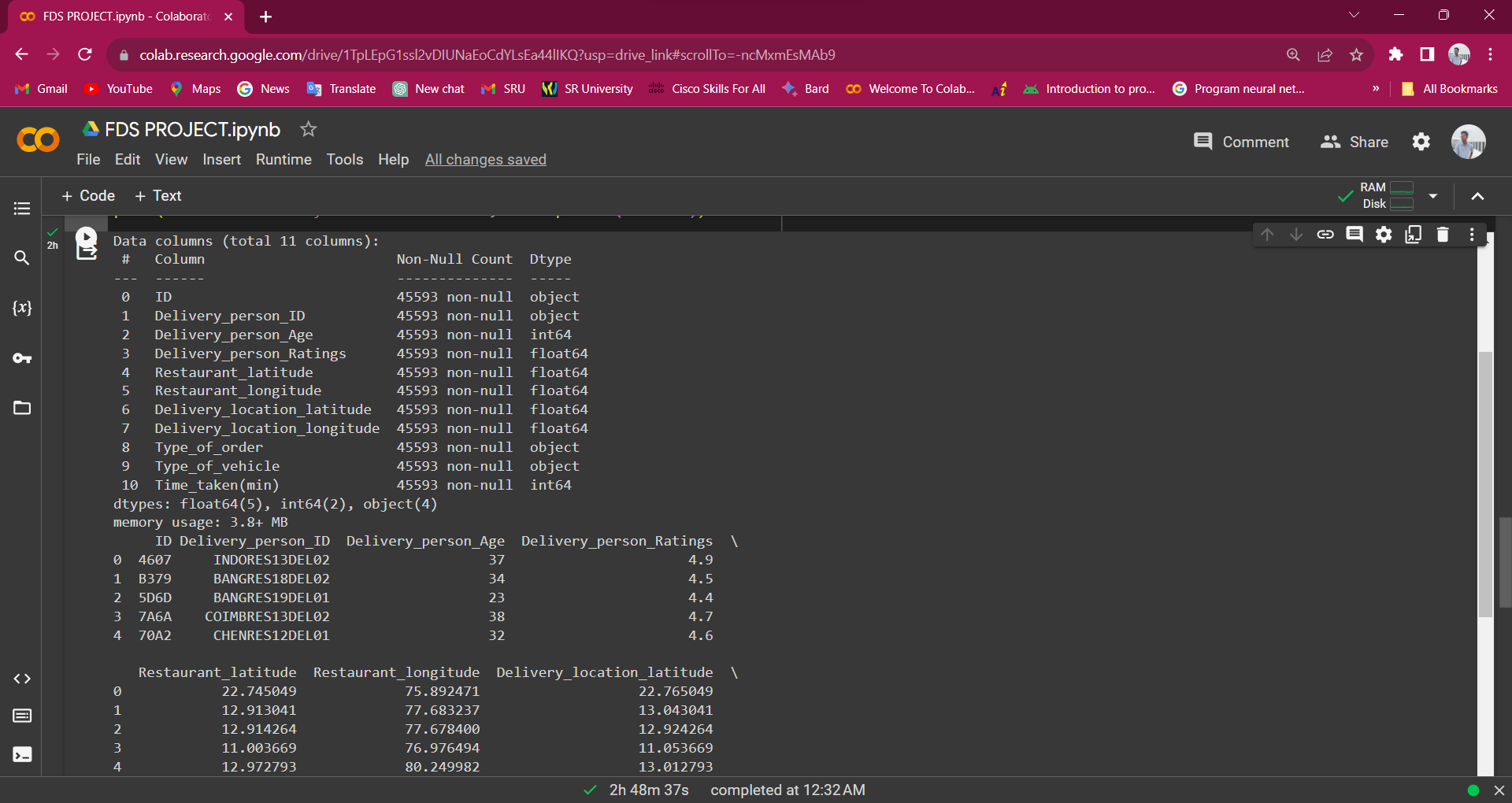


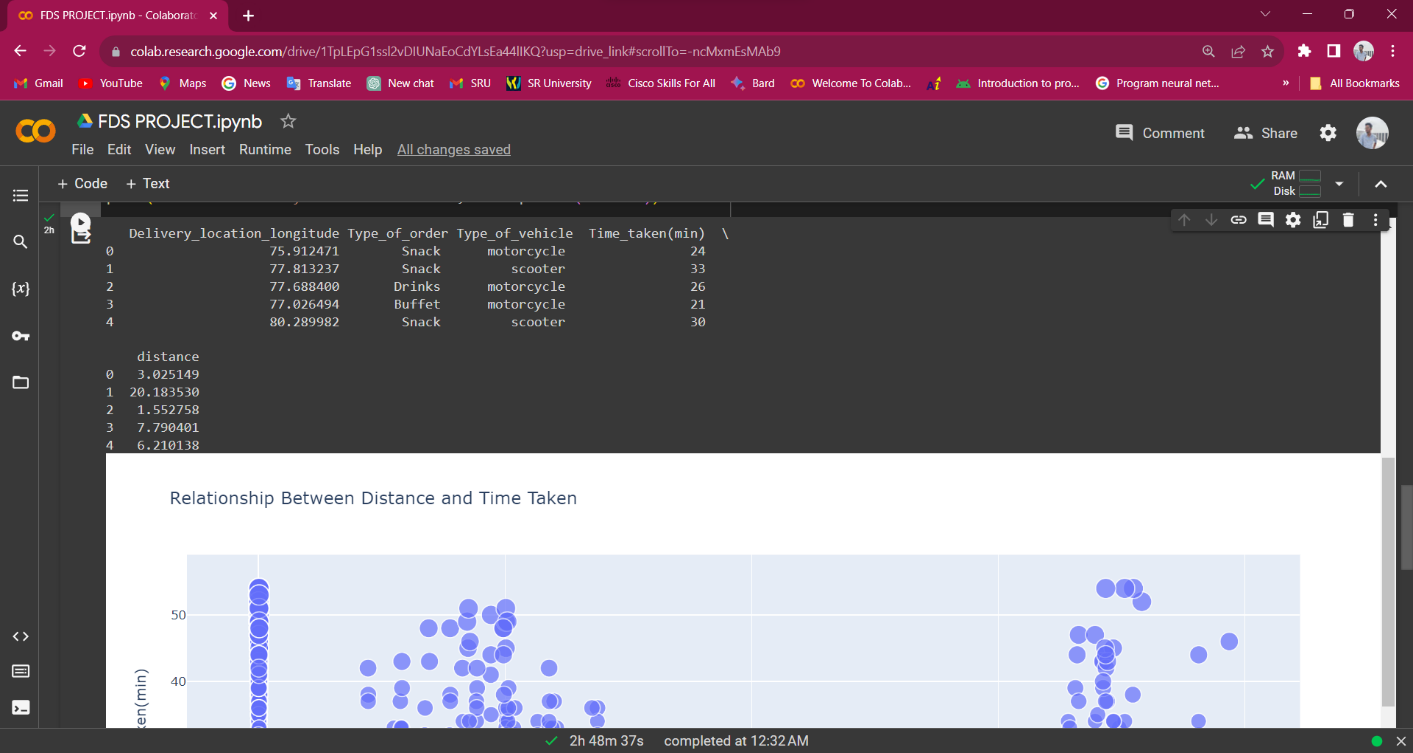


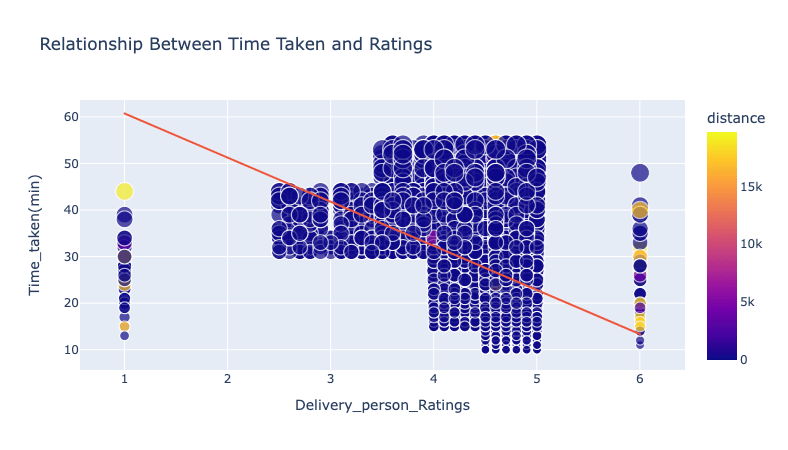
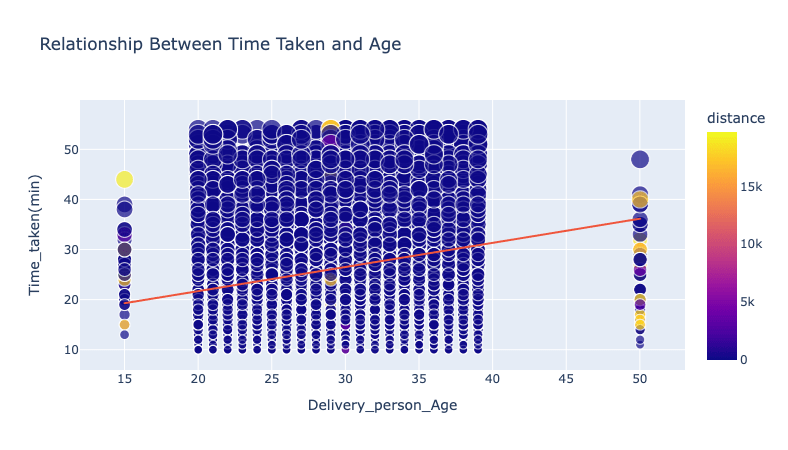
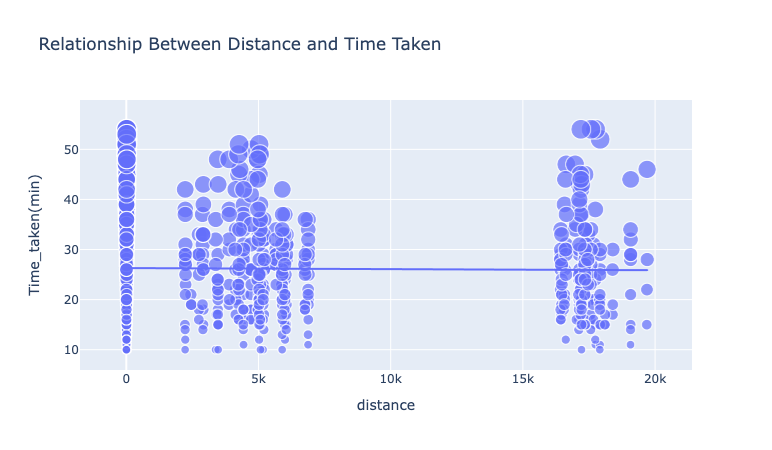


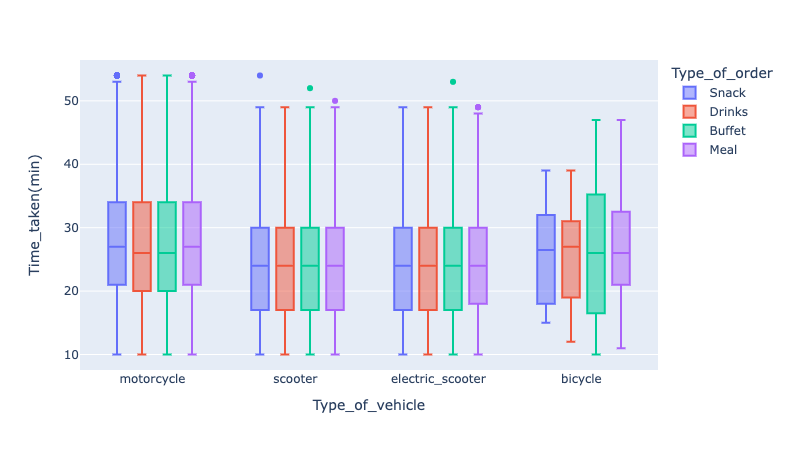


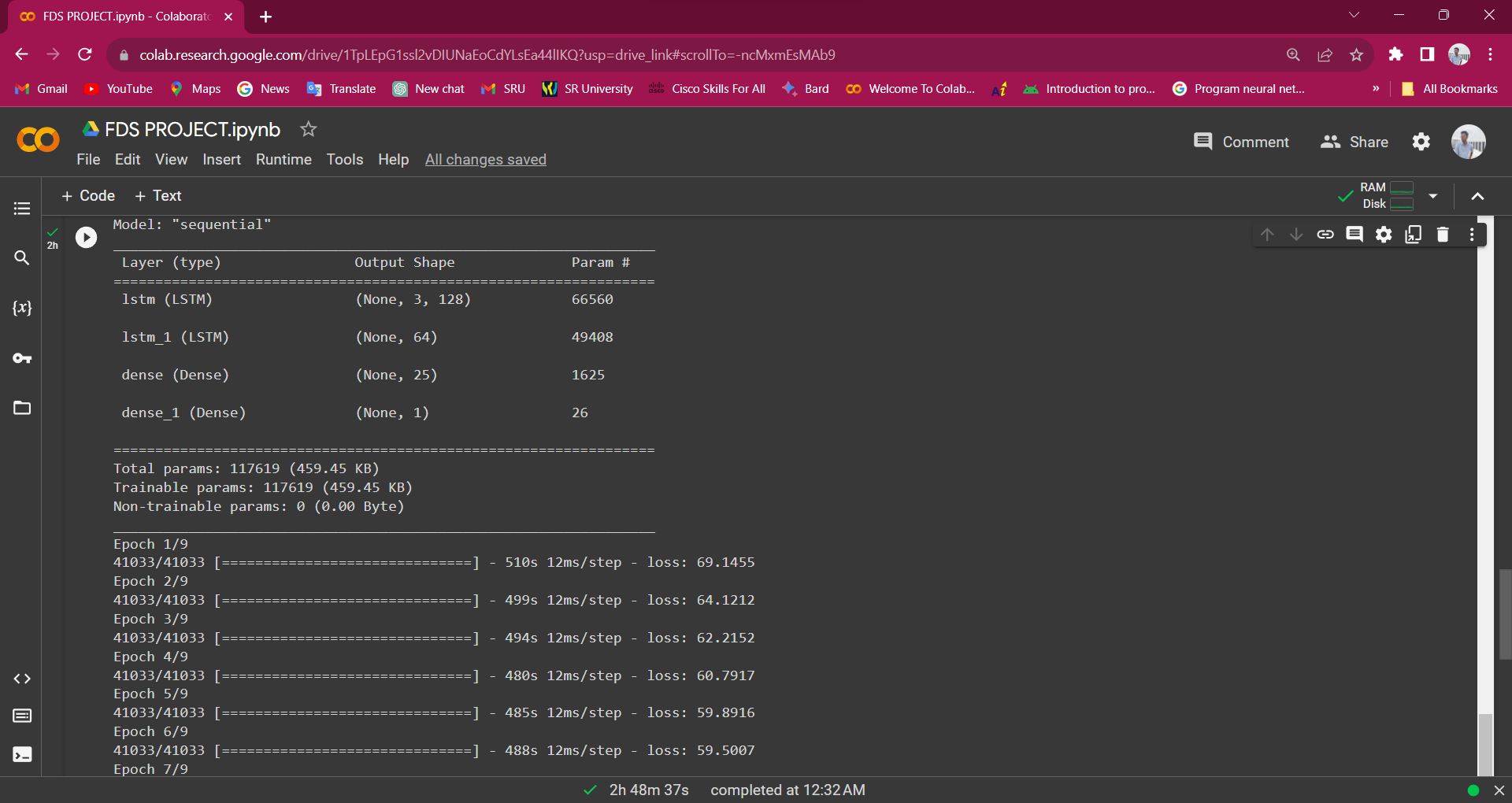


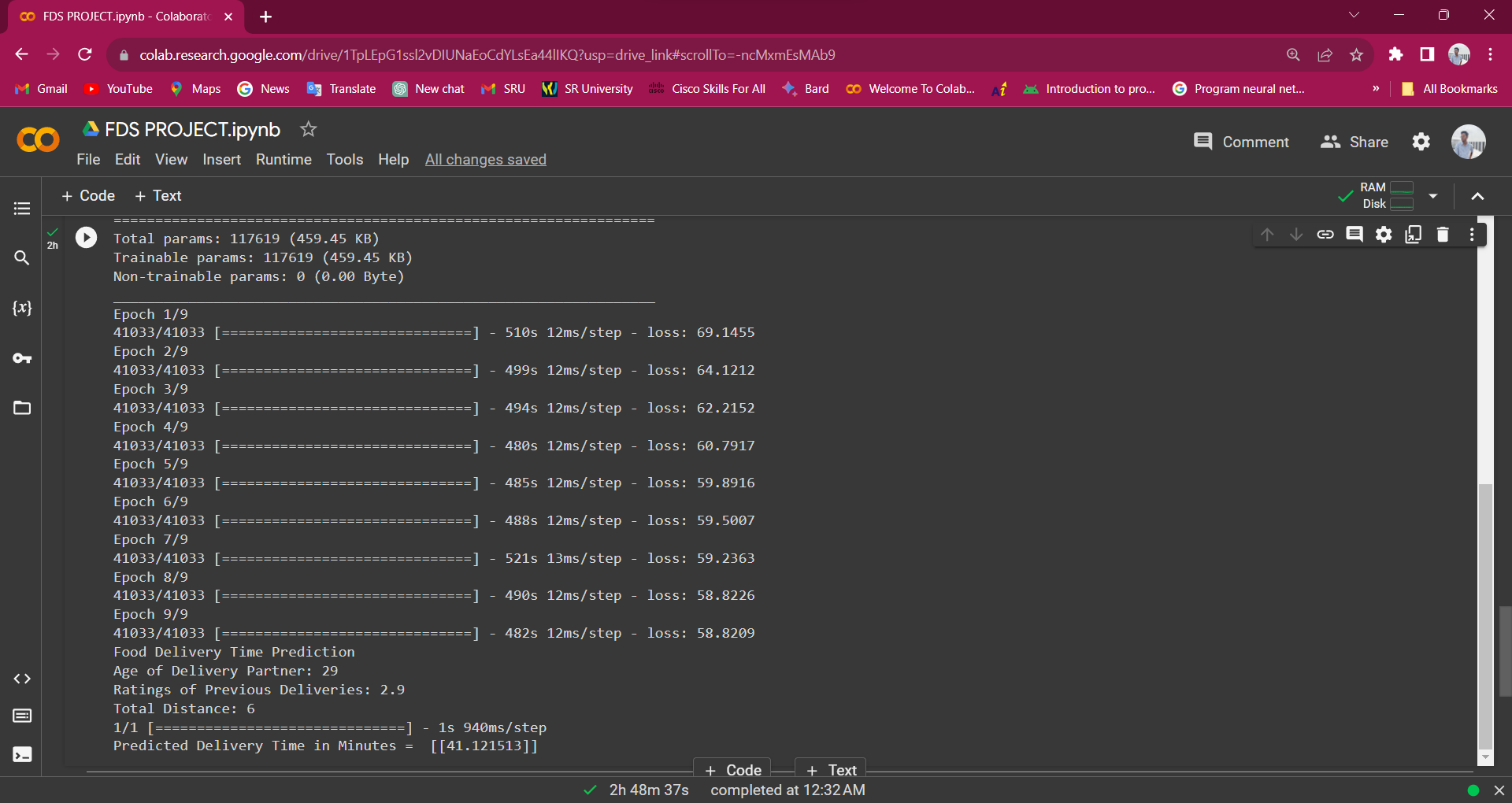












**Conclusion:**

In conclusion, the food delivery time prediction project demonstrates the use of data analysis, geospatial calculations, and machine learning techniques to estimate the time it takes for food to be delivered to customers. The project employs a Long Short-Term Memory (LSTM) neural network model and utilizes various Python libraries for data manipulation, visualization, and machine learning.

While the project provides a basic framework for time prediction.

In summary, while the project serves as a starting point for food delivery time prediction, a comprehensive and adaptable system for real-world use requires further development and consideration of factors like data quality, user experience, scalability, and regulatory compliance.

**Reference:**

Article-

[*https://thecleverprogrammer.com/2023/01/02/food-delivery-time-prediction-using-python/*](https://thecleverprogrammer.com/2023/01/02/food-delivery-time-prediction-using-python/)

Dataset-

[*https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset?select=train.csv*](https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset?select=train.csv)