

STREAMLIT APP FOR SALES PREDICTION USING EMBEDDED MACHINE LEARNING MODEL

**Deploy Machine Learning
model using Streamlit**



copyassignment.com

INTRODUCTION

Sales forecasting is a critical aspect of the retail industry as it informs management decisions regarding inventory, product placement, and marketing strategies. By understanding purchasing patterns and trends, accurate sales predictions can drive revenue growth. In the past, predicting sales was challenging, but with technological advancements, it has become easier. Machine learning and artificial intelligence tools enable industries to leverage vast amounts of diverse data to make accurate sales predictions.

WHAT IS STREAMLIT?

Streamlit is a Python library and app framework that enables users to build and share data applications quickly. It is designed for Machine Learning and Data Science projects and provides a customizable frontend. In this tutorial, we will create a basic web application that utilizes a loan default prediction ML model. The application will allow users to input various factors, and the model will predict whether or not the loan applicant can repay the loan.

Problem Objective & Machine Learning Formulation:

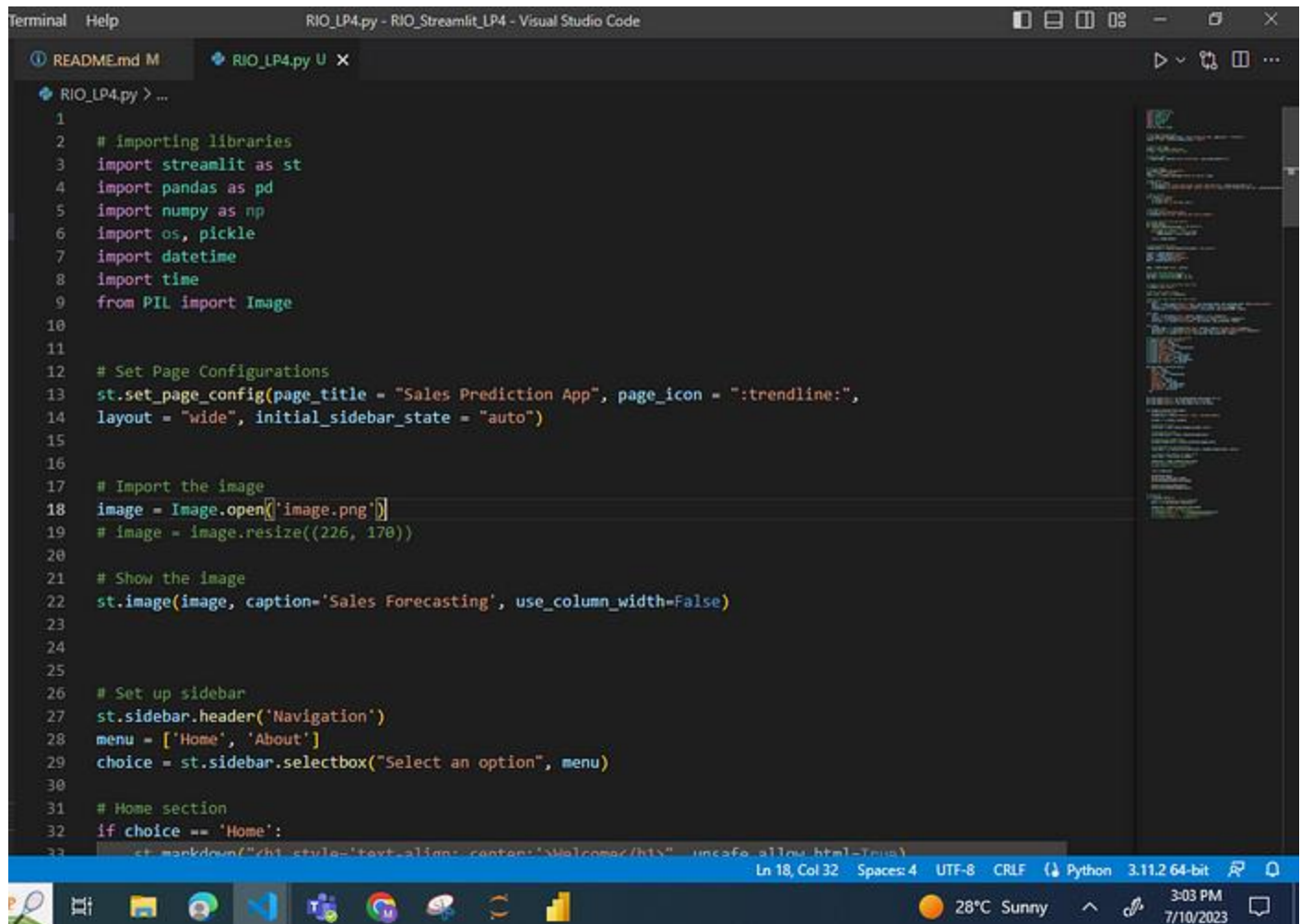
As described in the objective, the objective is to Configure Streamlit in environment and create a User Interface for regression model from [this](#) TIME SERIES REGRESSION project. The best-performing model of the analysis was used for the Streamlit app. It was therefore

required to create a functional Machine Learning App that will predict sales in real-time.

Creation of the Web App

The web application was developed with the help of Streamlit, an open-source Python framework. Streamlit enabled the easy creation of web apps and data visualizations, with an interactive and responsive user interface. It is designed on the backend, python was mainly used but supported with some HTML and CSS to aid the design of the interface.

In creating the app, a virtual environment was set up, and the necessary requirements were installed. Then, the app interface was designed.



```
1
2 # importing libraries
3 import streamlit as st
4 import pandas as pd
5 import numpy as np
6 import os, pickle
7 import datetime
8 import time
9 from PIL import Image
10
11
12 # Set Page Configurations
13 st.set_page_config(page_title = "Sales Prediction App", page_icon = ":trendline:",
14 layout = "wide", initial_sidebar_state = "auto")
15
16
17 # Import the image
18 image = Image.open('image.png')
19 # image = image.resize((226, 170))
20
21 # Show the image
22 st.image(image, caption='Sales Forecasting', use_column_width=False)
23
24
25
26 # Set up sidebar
27 st.sidebar.header('Navigation')
28 menu = ['Home', 'About']
29 choice = st.sidebar.selectbox("Select an option", menu)
30
31 # Home section
32 if choice == 'Home':
33     st.markdown("chi-style='text-align: center;'\nWelcome/(h1)" unsafe_allow_html=True)
```

Backend of the Web App and Packages installed

App Interface

To develop the app, essential libraries like pandas, numpy, scikit-learn, and Streamlit were imported, and the best model from the previous project, along with its components, was loaded. The loaded model will be used to predict data entered on the app's frontend.

Using streamlit functions, the input columns for the app were created. Widgets like `st.number_input`, `st.slider`, and `st.selectbox` were used to

specify the different inputs required for the model, enabling users to enter the correct information. The app's working principle is that when a user enters details on the interface, the entered variable will be converted into a pandas dataframe and taken through the pre-processing steps used to train the model. Then, the loaded model will make predictions on the processed dataframe from the interface.

To achieve this, a function was created to transform the variable from the user interface into a dataframe, which is then subjected to pre-processing.

```
Terminal  Help  RIO_LP4.py - RIO_Streamlit_LP4 - Visual Studio Code

① README.md M  + RIO_LP4.py U x

+ RIO_LP4.py > ...
63 encode = loaded_object["encoder"]
64 data = loaded_object["data"]
65
66
67 data_ = data.drop('sales', axis=1)
68
69 # Set Min and Max date interval
70 min_date = datetime.date(2000, 1, 1)
71 max_date = datetime.date(2009, 12, 31)
72
73
74 # Create a form for collecting input data
75 st.header("Input Data")
76
77 # Define the column layout
78 col1, col2, col3 = st.columns(3)
79
80 # Define the input fields for each column
81 with col1:
82     date = st.date_input("Select a date", min_value=min_date, max_value=max_date, key="my_date_picker")
83     family = st.selectbox("Family", options=(list( data_['family'].unique())))
84     transactions = st.slider("Transactions", min_value=1, max_value=10000, step=1)
85
86 with col2:
87     city = st.selectbox("City", options =(data_['city'].unique()))
88     cluster = st.selectbox("Cluster", options=(list( data_['cluster'].unique())))
89     store_nbr = st.slider("Store Number", min_value=1, max_value=100, step=1)
90
91 with col3:
92     holiday_type = st.selectbox("Day Type", options =(data_['holiday_type'].unique()))
93     onpromotion = st.selectbox("On Promotion", options=(list( data_['onpromotion'].unique())))
94     oil_price = st.slider("Oil Price", min_value=1, max_value=110, step=1)
95
```

Ln 18, Col 32 Spaces: 4 UTF-8 CRLF Python 3.11.2 64-bit

28°C Sunny 3:06 PM 7/10/2023

After Pre-processing & Sales Prediction

×

Navigation

Select an option

Home

Welcome

This is a simple sales prediction app.

Sales Prediction App

Select your features and click on Submit

Input Data

Select a date

2023/04/16

City

Salinas

Day Type

Holiday

Family

AUTOMOTIVE

Cluster

1

On Promotion

0

Transactions

1

Store Number

1

Old Price

1

Input Data Summary

Date: 2023-04-16

Family: AUTOMOTIVE

Transactions: 1

City: Salinas

A display of sale prediction

