Streamlit code

- We can build the interactive web application directly from python script we can create easily interactive dashboards, visualizations, and user interfaces & analysis of data.

```
import pickle
import sklearn
import pandas as pd
import streamlit as st
from streamlit_option_menu import option_menu
from streamlit_extras.colored_header import colored_header
import plotly.express as px
import os
from PIL import Image
import warnings
warnings.filterwarnings('ignore')
```

- Importing necessary libraries:
- pickle for saving and loading machine learning models.
- sklearn for machine learning algo and functionalities.
- streamlit for web application UI dashboard.
- extra colors creating colour header content.
- ploty for data visualization 3d plots.
- Modul for handling warnings in python.

```
df = pd.read_csv("final (1).csv")

class App:
    def model(self):
        # Setting Page Configuration
        st.set_page_config(page_title="Flat resale prediction | By Viswanathan", layout="wide")
        # Creating the option menu in the sidebar
        with st.sidebar:
            selected = option_menu(
                menu_title="FRP",
                options=["Home", "Explore Data","Flat resale_Prediction","Contact" ],
                icons=['house','barchart','graph-up-arrow', 'at'],
                menu_icon='alexa',
                default_index=0
            )
```

- Reading the final 1 csv file
- Creating the class an application called app which includes function model.
- Inside the model method streamlit page configuration and 4 options are setup.

Option Home

```
if selected == "Home":
    col1, col2, col3 = st.columns([4, 10, 2])
    with col2:
        st.markdown(
            "<h1 style='font-size: 40px;'><span style='color: cyan;'>Flat resale price</span><span style='color: white;'> Prediction</span> </h1>",
            unsafe_allow_html=True)
    st.subheader("Based on the properties of HDB flats in Singapore,the resale price can be predicted by data mining methods. This report will show how we did implement the knowledge
                 "practice EDA, data preprocessing and model training and complte the prediction of the resale price of HDB flats in Singapore")
    st.subheader('Skills take away From This Project:')
    st.subheader('Data Wrangling, EDA, Model Building, Model Deployment')
    st.subheader('Domain: RealEstate')
```

- Using the condition if User select home option from the side bar menu.
- It displays the information about the home page.

Option Explore data

```
if selected == "Explore Data":
    fl = st.file_uploader(":file_folder: Upload a file", type=(["csv", "txt", "xlsx", "xls"]))

    st.sidebar.header("Choose your filter: ")

    # Create for town
    town = st.sidebar.multiselect("Pick your town", df["town"].unique())
    if not town:
        df2 = df.copy()
    else:
        df2 = df[df["town"].isin(town)]
        # Create for street_name
    street_name = st.sidebar.multiselect("Pick the street_name", df2["street_name"].unique())
    if not street_name:
        df3 = df2.copy()
    else:
        df3 = df2[df2["street_name"].isin(street_name)]
```

- The condition blocks checks if user selected the explore data from sidebar menu.
- It displays the option for uploading the files and selecting the filters to explore the data.
- User can explore and analyze the datasets.

```
# Filter the data based on town, street_name
if not town and not street_name:
    filtered_df = df

elif not street_name:
    filtered_df = df[df["town"].isin(town)]

elif not town:
    filtered_df = df[df["street_name"].isin("street_name")]

elif street_name:
    filtered_df = df3[df["street_name"].isin("street_name")]

elif town:
    filtered_df = df3[df["town"].isin(town)]

elif town and street_name:
    filtered_df = df3[df["town"].isin(town) & df3["street_name"].isin(street_name)]

else:
    filtered_df = df3[df3["town"].isin(town) & df3["street_name"].isin(street_name)]
flat_type_df = filtered_df.groupby(by=["flat_type"], as_index=False)["resale_price"].sum()
```

- This blocks processes the filter conditions based on the selected town and street names.
- If user selected the filter it allows to data for future analysis and visualization based on user selected files.

.

```
col1, col2 = st.columns(2)
with col1:
    st.subheader("flat_type_ViewData")
fig = px.bar(flat_type_df, x="flat_type", y="resale_price",
             text=['${:,.2f}'.format(x) for x in flat_type_df["resale_price"]],
             template="seaborn")
st.plotly_chart(fig, use_container_width=True, height=200)

with col2:
    st.subheader("town_ViewData")
fig = px.pie(filtered_df, values="resale_price", names="town", hole=0.5)
fig.update_traces(text=filtered_df["town"], textposition="outside")
st.plotly_chart(fig, use_container_width=True)
```

- Visualizing using ploty_express of bar with flat_type and resale price.
- It creates a bar chart showing the total resale price for each flat type flat_type and a pie chart showing the distribution of resale prices across different towns.

```
cl1, cl2 = st.columns((2))
with cl1:
    with st.expander("flat_type wise resale_price"):
        st.write(flat_type_df.style.background_gradient(cmap="Blues"))
csv = flat_type_df.to_csv(index=False).encode('utf-8')
st.download_button("Download Data", data=csv, file_name="flat_type.csv", mime="text/csv",
                   help='Click here to download the data as a CSV file')
with cl2:
    with st.expander("town wise resale_price"):
        town = filtered_df.groupby(by="town", as_index=False)["resale_price"].sum()
st.write(town.style.background_gradient(cmap="Oranges"))
csv = town.to_csv(index=False).encode('utf-8')
```

- Displaying data frame table format and users can download the flat_type or resale price data by clicking the download button.

```
# Create a scatter plot
data1 = px.scatter(filtered_df, x="town", y="street_name", color="flat_type")
data1['layout'].update(title="flat_type in the street_name and town wise data using Scatter Plot.",
                       titlefont=dict(size=20), xaxis=dict(title="town", titlefont=dict(size=20)),
                       yaxis=dict(title="street_name", titlefont=dict(size=20)))
st.plotly_chart(data1, use_container_width=True)

with st.expander("Detailed Room Availability and resale_price View Data in the street_name"):
    st.write(filtered_df.iloc[:500, 1:20:2].style.background_gradient(cmap="Oranges"))
```

- scatter plot using Plotly Express to visualize data from the filtered_df DataFrame.
- The scatter plot represents the relationship between the town x-axis, street name y-axis and flat type color.

```
# Download orginal DataSet
csv = df.to_csv(index=False).encode('utf-8')
st.download_button('Download Data', data=csv, file_name="Data.csv", mime="text/csv")

import plotly.figure_factory as ff
st.subheader(":point_right: town wise flat_type and Resale_price")
with st.expander("Summary_Table"):
    df_sample = df[0:5][
        ["town", "street_name", "storey_range", "flat_type", "resale_price", "floor_area_sqm",
         "flat_model"]]
fig = ff.create_table(df_sample, colorscale="Cividis")
st.plotly_chart(fig, use_container_width=True)
```

- User can download the orginal csv data and displaying the data frame structure table of df columns.

Option Flat Resale Price Prediction

```
if selected == "Flat resale_Prediction":
    col1, col2, col3 = st.columns([4, 10, 2])
    with col2:
        st.markdown(
            "<h1 style='font-size: 80px;'><span style='color: cyan;'>Flat resale price</span><span style='color:Black;'> Prediction</span> </h1>",
            unsafe_allow_html=True)
    col1, col2, col3 = st.columns([4, 10, 5])
    with col2:
        colored_header(
            label="",
            description="",
            color_name="blue-green-70"
        )
```

- The condition blocks checks if user selected the explore data from sidebar menu.

```
# Start from options
col2.write("")
with col2:
    st.write("")
    st.write("")
    st.markdown(
        "<h1 style='font-size: 30px;'><span style='color: cyan;'>selling_month </h1>",
        unsafe_allow_html=True)
    selling_month = st.selectbox(' ', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
```

- It displays a selection box of drop down menu containing options for selecting the selling month variable.
- It stores the user value on selling month.

```
st.write("")
st.write("")
st.markdown("<h1 style='font-size: 30px;'><span style='color: cyan;'>selling_year </h1>",
            unsafe_allow_html=True)
selling_year = st.selectbox('    ', [1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000,
                                     2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011,
                                     2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022,
                                     2023, 2024])
```

- It displays a selection box of drop down menu containing options for selecting the selling year variable.
- It stores the user value on selling year.

```
st.markdown("<h1 style='font-size: 30px;'><span style='color: cyan;'>Town </h1>", unsafe_allow_html=True)
town_dict = dict(zip(data['town'].unique(), data['town_code'].unique()))
town_list = data['town'].unique()
town_key = st.selectbox('Town', options=town_list)
town = town_dict[town_key]
st.write("")
```

- Select a town from the list of unique towns in Data Frame
- Selected town is then mapped to its corresponding town code using a dictionary using drop down user can select.

```
st.markdown("<h1 style='font-size: 30px;'><span style='color: cyan;'>Store_min </h1>", unsafe_allow_html=True)
storey_min = st.number_input('', min_value=1, max_value=49, value=10,)
```

- It displays storey min and number input field where the user can enter a value within restricted range.

```
st.write("")
st.markdown("<h1 style='font-size: 30px;'><span style='color: cyan;'>Store_max </h1>",
            unsafe_allow_html=True)
storey_max = st.number_input('', min_value=3, max_value=51, value=12,)
st.write("")
```

- It displays storey max and number input field where the user can enter a value within restricted range.

```
st.markdown("<h1 style='font-size: 30px;'><span style='color: cyan;'>Floor_area_sqft </h1>", unsafe_allow_html=True)
floor_area_sqm = st.number_input('', min_value=28.0, max_value=307.0, value=31.0        ,)
```

- It displays floor area and number input field where the user can enter a value within restricted range.

```
st.markdown("<h1 style='font-size: 30px;'><span style='color: cyan;'>Flat_Model </h1>", unsafe_allow_html=True)
model_dict = dict(zip(data['flat_model'].unique(), data['flat_modelcode'].unique()))
model_list = data['flat_model'].unique()
flat_model = st.selectbox('Flat Model', options=model_list)
flat_modelcode = model_dict[flat_model]
```

- A flat_model from the list of unique flat in Data Frame the selected town is then mapped to its corresponding town code using a dictionary using drop down user can select.

```
st.markdown("<h1 style='font-size: 30px;'><span style='color: cyan;'>lease_commence_date</h1>", unsafe_allow_html=True)
lease_commence_date = st.selectbox('  ', [1977, 1976, 1978, 1979, 1984, 1980, 1985, 1981, 1982, 1986, 1972,
                                          1983, 1973, 1969, 1975, 1971, 1974, 1967, 1970, 1968, 1988, 1987,
                                          1989, 1990, 1992, 1993, 1994, 1991, 1995, 1996, 1997, 1998, 1999,
                                          2000, 2001, 1966, 2002, 2006, 2003, 2005, 2004, 2008, 2007, 2009,
                                          2010, 2012, 2011, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2022,
                                          2020])
```

- It displays the selection box of drop down menu for selecting the lease common column variable this input can be used as a feature for model.

```
predict_data = [selling_month, selling_year, town, storey_min, storey_max,
                floor_area_sqm,flat_modelcode,lease_commence_date]
```

- Stored the user values as list this data can be used as input for predicting the resale price of a property using a machine learning model.

```
    with open("S1_regression (2).pkl", 'rb') as f:
        model = pickle.load(f)
col1, col2, col3 = st.columns([10, 1, 10])
```

- Opening the file and use to load a pre-trained machine learning model from a binary file using the pickle module.
- Once the model is loaded, it can be used for making predictions on new data within the Streamlit application.

```
with col1:
    st.write("")
    if st.button('Process'):
        X = model.predict([predict_data])
        st.markdown(
            f"<h1 style='font-size: 30px;'><span style='color: cyan;'>Predicted Flat Resale: </span><span style='color: Black;'> {x[0]}</span> </h1>",
            unsafe_allow_html=True)
```

- When user select the process button the model will predict the value of resale price based on input data stored in the predict data.
- It predicted price will displayed

```
if selected == "Contact":
    col1, col2, col3 = st.columns([4, 10, 2])
    with col2:
        Name = (f'{"Name :"} {"Viswanathan"}')
        mail = (f'{"Mail :"} {"viswanathan9692@gmail.com"}')
        description = "An Aspiring DATA-SCIENTIST..!"
        social_media = {"GITHUB": "https://github.com/Viswanathan25"}
        st.header('Singapore Resale Flat resale_prices Predicting')
        st.subheader("The objective of this project is to develop a machine learning model and deploy it as a user-friendly web application that predicts the resale prices of flats i
        st.write("---")
        st.subheader(Name)
        st.subheader(mail)
        st.subheader(description)
        st.write("#")
        cols = st.columns(len(social_media))
        for index, (platform, link) in enumerate(social_media.items()):
            cols[index].write(f"[{platform}]({link})")
```

- if User select contact option the name and contact details will display on page and direct hyperlink was created.

```
# Object
Object = App()
Object.model()
```

- Calling the object app and function model.