In [1]:
```
from IPython.display import Image
Image(url="https://www.gvfinancebrokers.com.au/thumbnaillarge/Loan-Guidelines.jpg")
```

Out[1]:

This project builds a machine learning model to predict loan approval using logistic regression. We preprocess the data by cleaning column names, combining asset columns, and converting categorical data to numerical values. The data is then split into training and testing sets, and features are scaled. After training and evaluating the logistic regression model, we save the model and scaler. Finally, we deploy a Streamlit app to provide an interactive interface for users to input their data and receive loan approval predictions.

“Loan Approval Prediction Model”

In [2]:
```
# Importing Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
import pickle as pk
```

In [3]:
```
# Load dataset
data = pd.read_csv("C:\\Users\\sahil\\Loan_Approval_Prediction\\loan_approval_datas
```

In [4]:
```
print(data)
```

```
      loan_id  no_of_dependents      education self_employed  income_annum  \
0           1                 2       Graduate            No       9600000
1           2                 0   Not Graduate           Yes       4100000
2           3                 3       Graduate            No       9100000
3           4                 3       Graduate            No       8200000
4           5                 5   Not Graduate           Yes       9800000
...       ...               ...            ...           ...           ...
4264     4265                 5       Graduate           Yes       1000000
4265     4266                 0   Not Graduate           Yes       3300000
4266     4267                 2   Not Graduate            No       6500000
4267     4268                 1   Not Graduate            No       4100000
4268     4269                 1       Graduate            No       9200000

      loan_amount  loan_term  cibil_score  residential_assets_value  \
0        29900000         12          778                   2400000
1        12200000          8          417                   2700000
2        29700000         20          506                   7100000
3        30700000          8          467                  18200000
4        24200000         20          382                  12400000
...           ...        ...          ...                       ...
4264      2300000         12          317                   2800000
4265     11300000         20          559                   4200000
4266     23900000         18          457                   1200000
4267     12800000          8          780                   8200000
4268     29700000         10          607                  17800000

      commercial_assets_value  luxury_assets_value  bank_asset_value  \
0                    17600000             22700000           8000000
1                     2200000              8800000           3300000
2                     4500000             33300000          12800000
3                     3300000             23300000           7900000
4                     8200000             29400000           5000000
...                       ...                  ...               ...
4264                   500000              3300000            800000
4265                  2900000             11000000           1900000
4266                 12400000             18100000           7300000
4267                   700000             14100000           5800000
4268                 11800000             35700000          12000000

      loan_status
0        Approved
1        Rejected
2        Rejected
3        Rejected
4        Rejected
...           ...
4264     Rejected
4265     Approved
4266     Rejected
4267     Approved
4268     Approved

[4269 rows x 13 columns]
```

In [5]:

```
data.head()
```

Out[5]:

	loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term
0	1	2	Graduate	No	9600000	29900000	12
1	2	0	Not Graduate	Yes	4100000	12200000	8
2	3	3	Graduate	No	9100000	29700000	20
3	4	3	Graduate	No	8200000	30700000	8
4	5	5	Not Graduate	Yes	9800000	24200000	20

In [6]:
```
data.isnull().any()
```

Out[6]:
```
loan_id                     False
 no_of_dependents           False
 education                  False

 self_employed              False
 income_annum               False
 loan_amount                False
 loan_term                  False
 cibil_score                False
 residential_assets_value   False
 commercial_assets_value    False
 luxury_assets_value        False
 bank_asset_value           False
 loan_status                False
dtype: bool
```

In [7]:
```
# Drop the loan_id column as it is not needed
data.drop(columns=['loan_id'], inplace=True)
```

In [8]:
```
# Clean column names by stripping leading and trailing spaces
data.columns = data.columns.str.strip()
```

In [9]:
```
# Combining various asset columns into a single 'Assets' column
data['Assets'] = (data['residential_assets_value'] +
                  data['commercial_assets_value'] +
                  data['luxury_assets_value'] +
                  data['bank_asset_value'])
```

In [10]:
```
# Dropping the individual asset columns as they are now combined
data.drop(columns=['residential_assets_value', 'commercial_assets_value',
                   'luxury_assets_value', 'bank_asset_value'], inplace=True)
```

In [11]:
```
# Clean and convert categorical data
def clean_data(st):
    return st.strip()

data['education'] = data['education'].apply(clean_data).replace({'Graduate': 1, 'No
data['self_employed'] = data['self_employed'].apply(clean_data).replace({'No': 0, '
data['loan_status'] = data['loan_status'].apply(clean_data).replace({'Approved': 1,
```

In [12]:
```
# Splitting data into input and output
input_data = data.drop(columns=['loan_status'])
output_data = data['loan_status']
```

In [13]:
```
# Splitting data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(input_data, output_data, test_s
```

In [14]:
```
# Scaling the data
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [15]:
```
# Training the logistic regression model
model = LogisticRegression(random_state=42)
model.fit(x_train_scaled, y_train)
```

Out[15]:
```
▾ LogisticRegression
LogisticRegression(random_state=42)
```

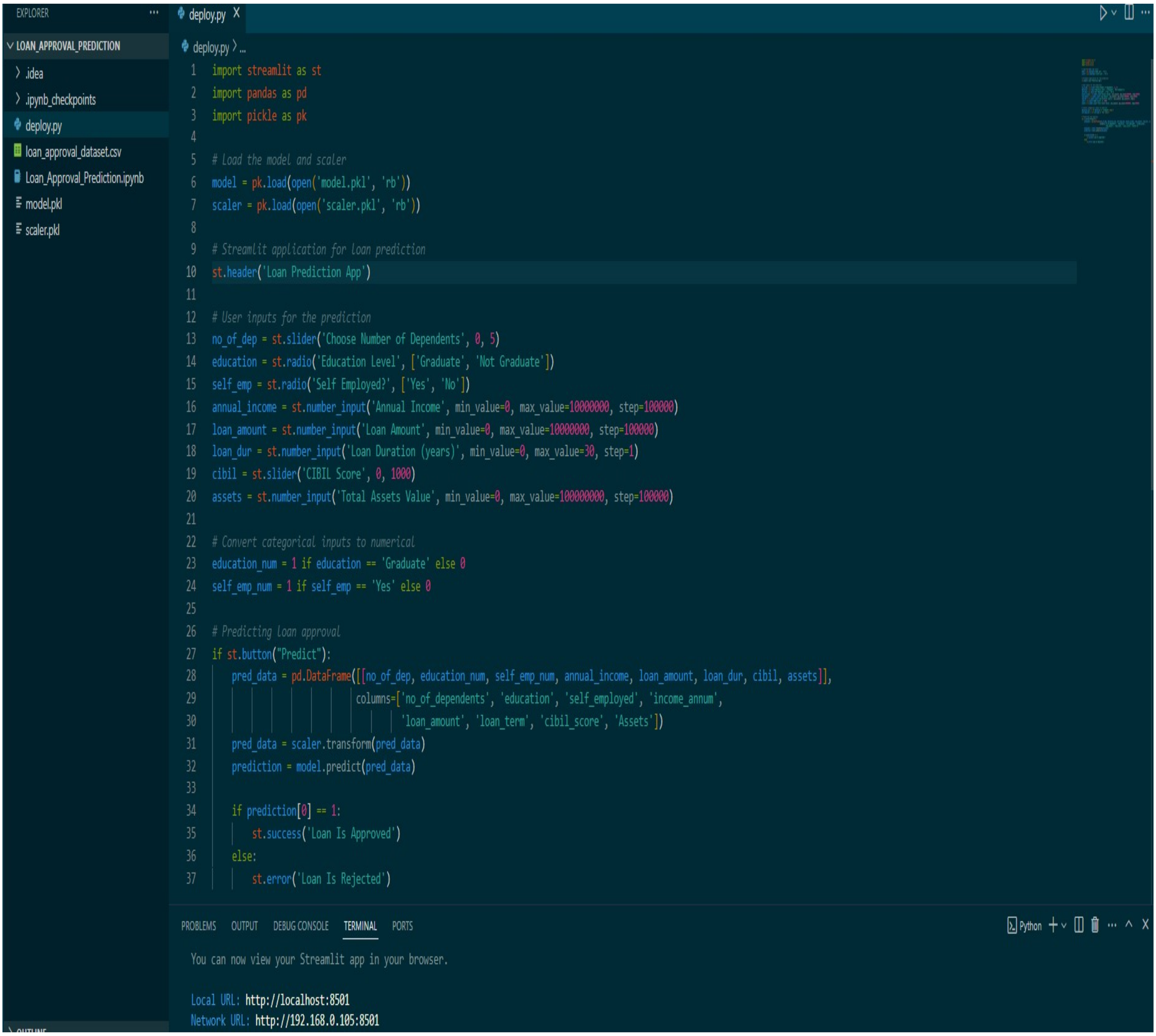
In [16]:
```
# Evaluating the model
score = model.score(x_test_scaled, y_test)
print(f"Model Accuracy: {score}")
```

```
Model Accuracy: 0.905152224824356
```

In [17]:
```
# Saving the model and scaler
with open('model.pkl', 'wb') as model_file:
    pk.dump(model, model_file)
with open('scaler.pkl', 'wb') as scaler_file:
    pk.dump(scaler, scaler_file)
```

In [ ]:

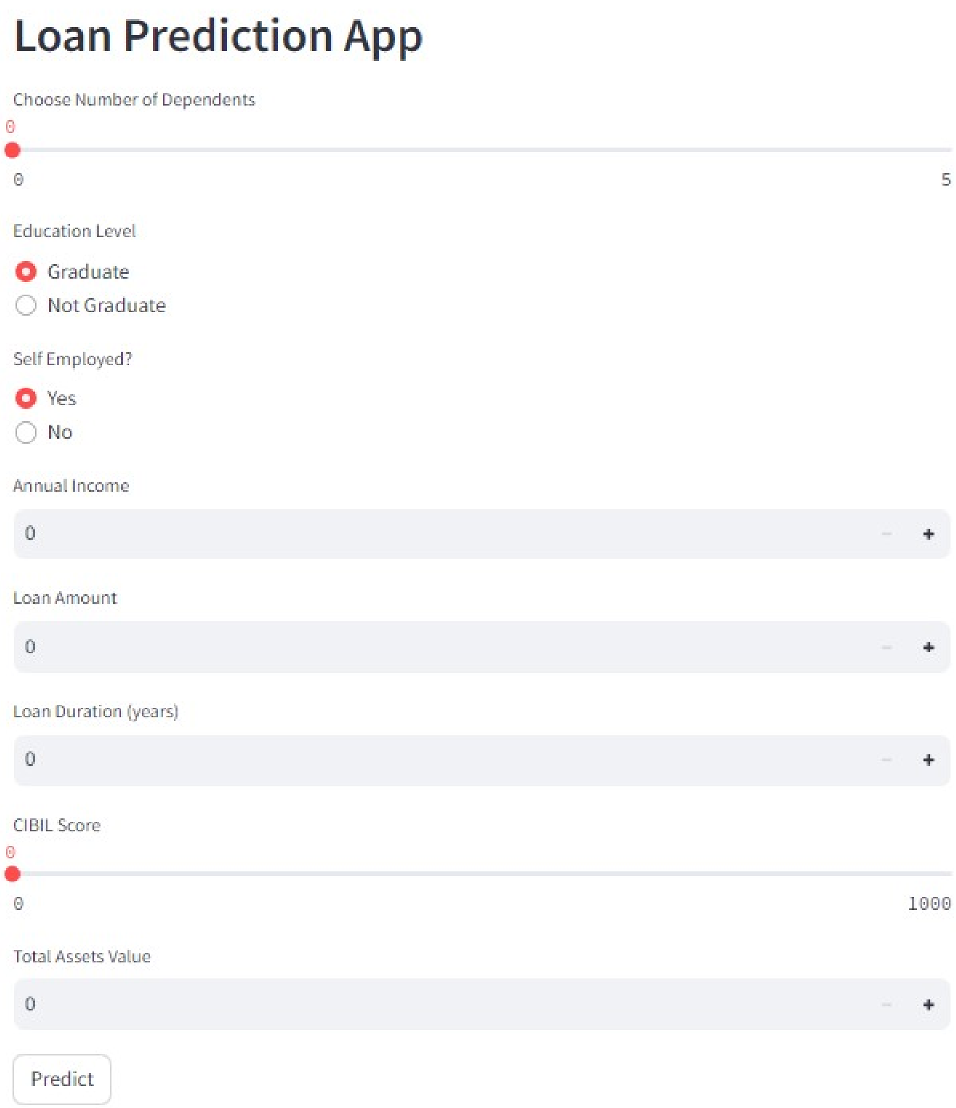
To deploy the trained model as a web application using Streamlit, allowing users to interact with the model and get predictions based on their input.



```python
import streamlit as st
import pandas as pd
import pickle as pk

# Load the model and scaler
model = pk.load(open('model.pkl', 'rb'))
scaler = pk.load(open('scaler.pkl', 'rb'))

# Streamlit application for loan prediction
st.header('Loan Prediction App')

# User inputs for the prediction
no_of_dep = st.slider('Choose Number of Dependents', 0, 5)
education = st.radio('Education Level', ['Graduate', 'Not Graduate'])
self_emp = st.radio('Self Employed?', ['Yes', 'No'])
annual_income = st.number_input('Annual Income', min_value=0, max_value=10000000, step=100000)
loan_amount = st.number_input('Loan Amount', min_value=0, max_value=10000000, step=100000)
loan_dur = st.number_input('Loan Duration (years)', min_value=0, max_value=30, step=1)
cibil = st.slider('CIBIL Score', 0, 1000)
assets = st.number_input('Total Assets Value', min_value=0, max_value=100000000, step=100000)

# Convert categorical inputs to numerical
education_num = 1 if education == 'Graduate' else 0
self_emp_num = 1 if self_emp == 'Yes' else 0

# Predicting loan approval
if st.button("Predict"):
    pred_data = pd.DataFrame([[no_of_dep, education_num, self_emp_num, annual_income, loan_amount, loan_dur, cibil, assets]],
                             columns=['no_of_dependents', 'education', 'self_employed', 'income_annum',
                                      'loan_amount', 'loan_term', 'cibil_score', 'Assets'])
    pred_data = scaler.transform(pred_data)
    prediction = model.predict(pred_data)

    if prediction[0] == 1:
        st.success('Loan Is Approved')
    else:
        st.error('Loan Is Rejected')
```

```
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.0.105:8501
```

The Streamlit application provides an intuitive and user-friendly interface for users to input their data and receive loan approval predictions.



Loan Prediction App

Choose Number of Dependents

2

0 5

Education Level

- (•) Graduate
- ( ) Not Graduate

Self Employed?

- (•) Yes
- ( ) No

Annual Income

1000000 - +

Loan Amount

200000 - +

Loan Duration (years)

5 - +

CIBIL Score

518

0 1000

Total Assets Value

3211456 - +

Predict

Loan Is Approved

Loan Prediction App

Choose Number of Dependents

4

0 5

Education Level

- ○ Graduate
- ● Not Graduate

Self Employed?

- ● Yes
- ○ No

Annual Income

10000 − +

Loan Amount

2000000 − +

Loan Duration (years)

7 − +

CIBIL Score

71

0 1000

Total Assets Value

65432 − +

Predict

Loan Is Rejected