loanpredictionapp.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

..
sample_data
loan_approval_dataset.csv
loan_model.pkl
my_streamlit_app.py

Disk 76.85 GB available

+ Code + Text

RAM Disk Gemini

```
[22] import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
     from sklearn.model_selection import train_test_split
```

```
[23] df = pd.read_csv("/content/loan_approval_dataset.csv")
     df.head(3)
```

	loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	cibil_score	residential_assets
0	1	2	Graduate	No	9600000	29900000	12	778	2
1	2	0	Not Graduate	Yes	4100000	12200000	8	417	2
2	3	3	Graduate	No	9100000	29700000	20	506	7

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
[24] print("Column names in the DataFrame before stripping spaces:")
     print(df.columns.tolist())
```

```
Column names in the DataFrame before stripping spaces:
['loan_id', ' no_of_dependents', ' education', ' self_employed', ' income_annum', ' loan_amount', ' loan_term', ' cibil_sc
```

```
df.columns = df.columns.str.strip()
print("Column names after stripping spaces:")
print(df.columns.tolist())
```

```
Column names after stripping spaces:
['loan_id', 'no_of_dependents', 'education', 'self_employed', 'income_annum', 'loan_amount', 'loan_term', 'cibil_score', '
```

```
[26] if 'loan_id' in df.columns:
         df = df.drop(['loan_id'], axis=1)
```

```
[27] from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()

     df["education"] = le.fit_transform(df["education"])
     df["self_employed"] = le.fit_transform(df["self_employed"])
     df["loan_status"] = le.fit_transform(df["loan_status"])
```

```
[28] df.fillna(df.median(), inplace=True)
```

```
from sklearn.preprocessing import StandardScaler
numerical_features = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
if "loan_status" in numerical_features:
    numerical_features.remove("loan_status")
scaler = StandardScaler()
df[numerical_features] = scaler.fit_transform(df[numerical_features])
```

```
[31]
     x = df.drop(["loan_status"],axis=1)
     y = df["loan_status"]

     from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.1)
```

```
[32] from sklearn.linear_model import LogisticRegression
     reg = LogisticRegression()
     reg.fit(x_train,y_train)
     regpred = reg.predict(x_test)
```

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
print(classification_report(y_test,regpred))
print("Accuracy of Logistic Regression is : ",accuracy_score(y_test,regpred)*100)
```

```
              precision    recall  f1-score   support

           0       0.91      0.93      0.92       256
           1       0.89      0.87      0.88       171

    accuracy                           0.90       427
   macro avg       0.90      0.90      0.90       427
weighted avg       0.90      0.90      0.90       427

Accuracy of Logistic Regression is :  90.1639344262295
```

```
[34] from sklearn.neighbors import KNeighborsClassifier
     knn = KNeighborsClassifier(n_neighbors=3)
     knn.fit(x_train, y_train)
     y_pred = knn.predict(x_test)
     print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
     print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Confusion Matrix:
 [[224  32]
 [ 26 145]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.88      0.89       256
           1       0.82      0.85      0.83       171

    accuracy                           0.86       427
   macro avg       0.86      0.86      0.86       427
weighted avg       0.87      0.86      0.86       427
```

```
[35] import pickle
```

```
[36] filename = 'loan_model.pkl'
     pickle.dump(reg, open(filename, 'wb'))
```

```
[37] load_model = pickle.load(open(filename, 'rb'))
```

```
[38] !pip install -q streamlit
```

```
[39] !wget -q -O - ipv4.icanhazip.com
```

```
104.199.162.99
```

```
%%writefile my_streamlit_app.py
import streamlit as st
import pickle
import numpy as np

filename = 'loan_model.pkl'
loaded_model = pickle.load(open(filename, 'rb'))

st.title('Loan Approval Prediction App')
st.header('Please enter the following details:')

gender = st.selectbox('Gender', ['Male', 'Female'])
married = st.selectbox('Marital Status', ['Yes', 'No'])
dependents = st.selectbox('Number of Dependents', ['0', '1', '2', '3+'])
education = st.selectbox('Education Level', ['Graduate', 'Not Graduate'])
self_employed = st.selectbox('Self-Employed', ['Yes', 'No'])
applicant_income = st.number_input('Applicant Income')
coapplicant_income = st.number_input('Co-Applicant Income')
loan_amount = st.number_input('Loan Amount')
loan_term = st.number_input('Loan Term (in months)')
credit_history = st.selectbox('Credit History', ['0', '1'])
property_area = st.selectbox('Property Area', ['Urban', 'Semiurban', 'Rural'])

gender_encoded = 1 if gender == 'Male' else 0
married_encoded = 1 if married == 'Yes' else 0
dependents_encoded = 3 if dependents == '3+' else int(dependents)
education_encoded = 1 if education == 'Graduate' else 0
self_employed_encoded = 1 if self_employed == 'Yes' else 0
credit_history_encoded = int(credit_history)
property_area_encoded = {'Urban': 0, 'Rural': 1, 'Semiurban': 2}[property_area]

input_data = [
    gender_encoded, married_encoded, dependents_encoded, education_encoded,
    self_employed_encoded, applicant_income, coapplicant_income, loan_amount,
    loan_term, credit_history_encoded, property_area_encoded
]

try:
    st.write("Collected input data:", input_data)
    input_values = [float(value) for value in input_data]
    st.write("Converted input values:", input_values)
    if st.button('Predict Loan Approval'):
        input_values = np.array(input_values).reshape(1, -1)
        result = loaded_model.predict(input_values)
        if result[0] == 0:
            st.write('Sorry, your loan application is not approved.')
        else:
            st.write('Congratulations! Your loan application is approved.')
except ValueError as e:
    st.write(f"Value error occurred: {e}")
except KeyError as e:
    st.write(f"Key error occurred: {e} - ensure all categorical values are correctly mapped.")
except Exception as e:
    st.write(f"An unexpected error occurred: {e}")
```

```
Overwriting my_streamlit_app.py
```

```
!streamlit run my_streamlit_app.py & npx localtunnel --port 8501
```

```
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.


  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.28.0.12:8501
  External URL: http://104.199.162.99:8501

your url is: https://public-friends-fix.loca.lt
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:465: UserWarning: X does not have valid feature names, but Logisti
  warnings.warn(
```

```
[21] Start coding or generate with AI.
```

0s

Colab paid products - Cancel contracts here

Executing (6m 2s) <cell line: 1> › system() › _system_compat() › _run_command() › _monitor_process() › _poll_process()