**🔷 خطة المشروع: تحليل بيانات القروض وبناء نموذج تنبؤ**

**🟢 المرحلة 1: فهم البيانات**

* تحميل ملف البيانات loan\_prediction.csv.
* استعراض الأعمدة والبيانات باستخدام:

df.info()

df.describe(include='all')

* فهم طبيعة كل عمود (هل هو رقمي؟ نوعي؟ هدف التنبؤ؟).

|  |  |  |  |
| --- | --- | --- | --- |
| العمود | نوع المتغير | الوصف | ملاحظات |
| Loan\_ID | معرف (غير رقمي) | رقم تعريفي فريد لكل طلب قرض | لا يستخدم في النمذجة – نحذفه |
| Gender | نوعي (Categorical) | جنس المتقدم (Male / Female) | يحوّل إلى أرقام |
| Married | نوعي (Categorical) | هل المتقدم متزوج؟ (Yes / No) | يحوّل إلى أرقام |
| Dependents | نوعي/عددي مختلط | عدد المعالين (0, 1, 2, 3+) | يجب تحويل "3+" إلى عدد صحيح |
| Education | نوعي (Categorical) | هل المتقدم خريج جامعة؟ (Graduate / Not Graduate) | يحوّل إلى أرقام |
| Self\_Employed | نوعي (Categorical) | هل يعمل لحسابه الخاص؟ (Yes / No) | يحوّل إلى أرقام |
| ApplicantIncome | رقمي (Numerical) | دخل مقدم الطلب | مدخل مهم للنموذج |
| CoapplicantIncome | رقمي (Numerical) | دخل الشريك إن وجد | مدخل مهم |
| LoanAmount | رقمي (Numerical) | مبلغ القرض المطلوب | فيه قيم ناقصة - نعالجها |
| Loan\_Amount\_Term | رقمي (Numerical) | مدة سداد القرض بالأشهر (مثلاً 360 = 30 سنة) | فيه قيم ناقصة - نعالجها |
| Credit\_History | نوعي عددي (Binary) | هل لديه سجل ائتماني جيد؟ (1.0 = نعم، 0.0 = لا) | مهم جدًا للنموذج |
| Property\_Area | نوعي (Categorical) | منطقة العقار (Urban / Semiurban / Rural) | يحوّل إلى أرقام |
| Loan\_Status | نوعي ثنائي (Binary) | هل تمت الموافقة على القرض؟ (Y/N) | هدف التنبؤ (target) |

**🟢 المرحلة 2: التحليل الاستكشافي للبيانات (EDA)**

**2.1 استكشاف القيم المفقودة**

* معرفة الأعمدة التي تحتوي على قيم ناقصة:

df.isnull().sum()

**2.2 تحليل الأعمدة النوعية (Categorical Variables)**

* عرض التكرارات لكل فئة:

for col in ['Gender', 'Married', 'Education', 'Self\_Employed', 'Property\_Area', 'Loan\_Status']:

print(df[col].value\_counts())

df[col].value\_counts().plot(kind='bar', title=col)

**2.3 تحليل الأعمدة الرقمية (Numerical Variables)**

* رسم histogram لفهم توزيع الدخل والقرض:

df[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount']].hist(bins=20, figsize=(10, 6))

**2.4 فهم العلاقة بين المتغيرات وهدف التنبؤ**

* باستخدام boxplot:

import seaborn as sns

sns.boxplot(x='Loan\_Status', y='LoanAmount', data=df)

sns.boxplot(x='Loan\_Status', y='ApplicantIncome', data=df)

**🟢 المرحلة 3: تنظيف البيانات (Data Cleaning)**

**3.1 حذف الأعمدة غير المفيدة**

* حذف عمود Loan\_ID:

df.drop('Loan\_ID', axis=1, inplace=True)

**3.2 معالجة القيم المفقودة**

**LoanAmount:**

* الخطوة الأولى: عرض الإحصائيات العامة:

df['LoanAmount'].describe()

* الخطوة الثانية: رسم boxplot لاكتشاف القيم الشاذة:

sns.boxplot(x=df['LoanAmount'])

* الخطوة الثالثة: تعويض القيم المفقودة:
  + إذا التوزيع طبيعي:

df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)

* + إذا توجد قيم شاذة كثيرة:

df['LoanAmount'].fillna(df['LoanAmount'].median(), inplace=True)

**➤ أعمدة أخرى:**

* القيم النوعية (مثل Gender, Married...): تعويض بـ **القيمة الأكثر تكرارًا**:

df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)

**🟢 المرحلة 4: تحويل البيانات (Data Transformation)**

**4.1 ترميز المتغيرات النوعية**

* خيار 1: Label Encoding (إذا الفئات مرتبة أو binary):

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df['Gender'] = le.fit\_transform(df['Gender'])

* خيار 2: One-Hot Encoding (لتحويل الفئات إلى أعمدة):

df = pd.get\_dummies(df, drop\_first=True)

**4.2 تحجيم البيانات (اختياري لبعض النماذج)**

* باستخدام:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df\_scaled = scaler.fit\_transform(df)

**🟢 المرحلة 5: بناء النموذج (Model Building)**

* تقسيم البيانات إلى تدريب واختبار:

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

X = df.drop('Loan\_Status', axis=1)

y = df['Loan\_Status']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

* تجربة أكثر من نموذج:
  + Logistic Regression
  + Random Forest
* تدريب النموذج وتقييمه:

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

from sklearn.metrics import accuracy\_score, classification\_report

print(accuracy\_score(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))