

Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

**RAPORT**

**Lucrarea de laborator nr.4**

**la Inteligența Artificială**

*Tema: Rețele neuronale artificiale*

Grupa academică: TI-211  
A efectuat: Popa Cătălin

**A verificat: Mariana Rusu**

Chișinău 2024

Mai întâi am importat datele din csv și am afișat descrierea.

data\_info = pd.read\_csv('lending\_club\_info.csv', index\_col='LoanStatNew')  
print(data\_info.loc['revol\_util']['Description'])  
  
def feat\_info(col\_name):  
 print(data\_info.loc[col\_name]['Description'])

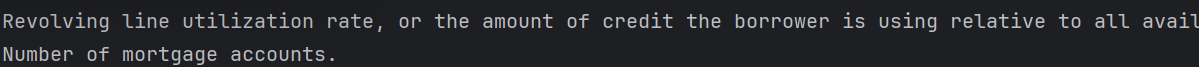


Figura 1 – Importarea datelor

file\_path = 'lending\_club\_loan\_two.csv'  
df = pd.read\_csv(file\_path)

Afișarea informației despre datele care vor fi folosite.

df.info()

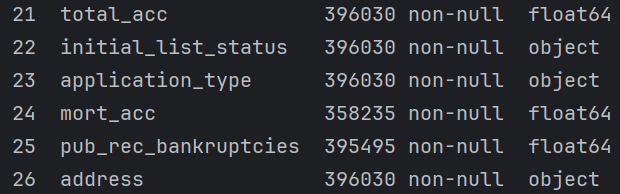


Figura 2 – Informașii de datele din csv.

**Secțiunea 1 : Analiza exploratorie a datelor**

**Obiectiv general**

Obținerea unei înțelegeri pentru variabile importante, vizualizare statistici statistici.

Crearea unui countplot.

def In9():  
 sns.countplot(x="loan\_status", data=df, palette={"Fully Paid": "blue", "Charged Off": "orange"}, legend=False)  
 plt.show()

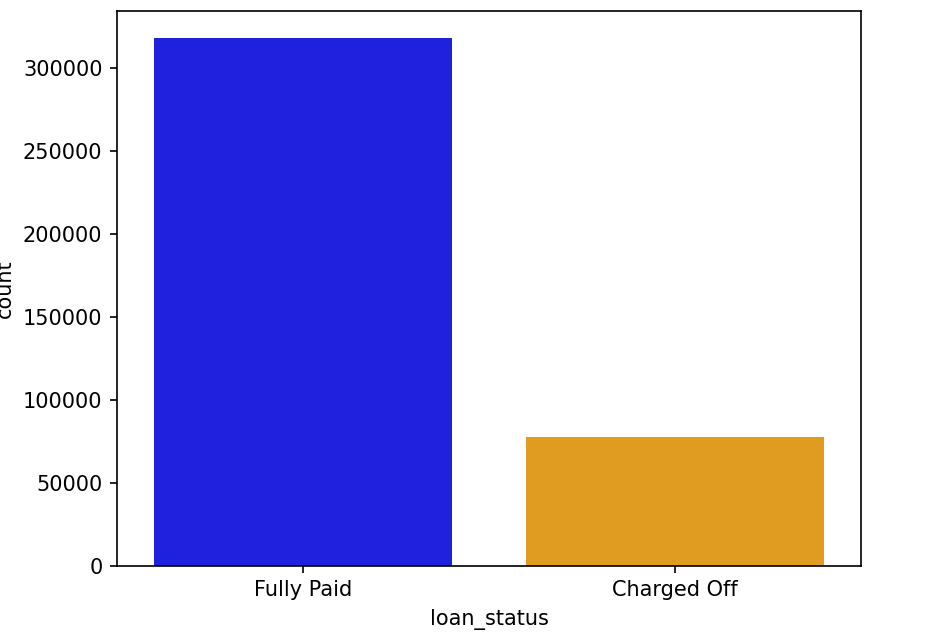


Figura 3 – Afisare countplot.

Crearea histogramei pentru coloana loan\_amnt.

def In10():  
 plt.figure(figsize=(12,6))  
 sns.distplot(df['loan\_amnt'], kde=False, bins=60)  
 plt.show()

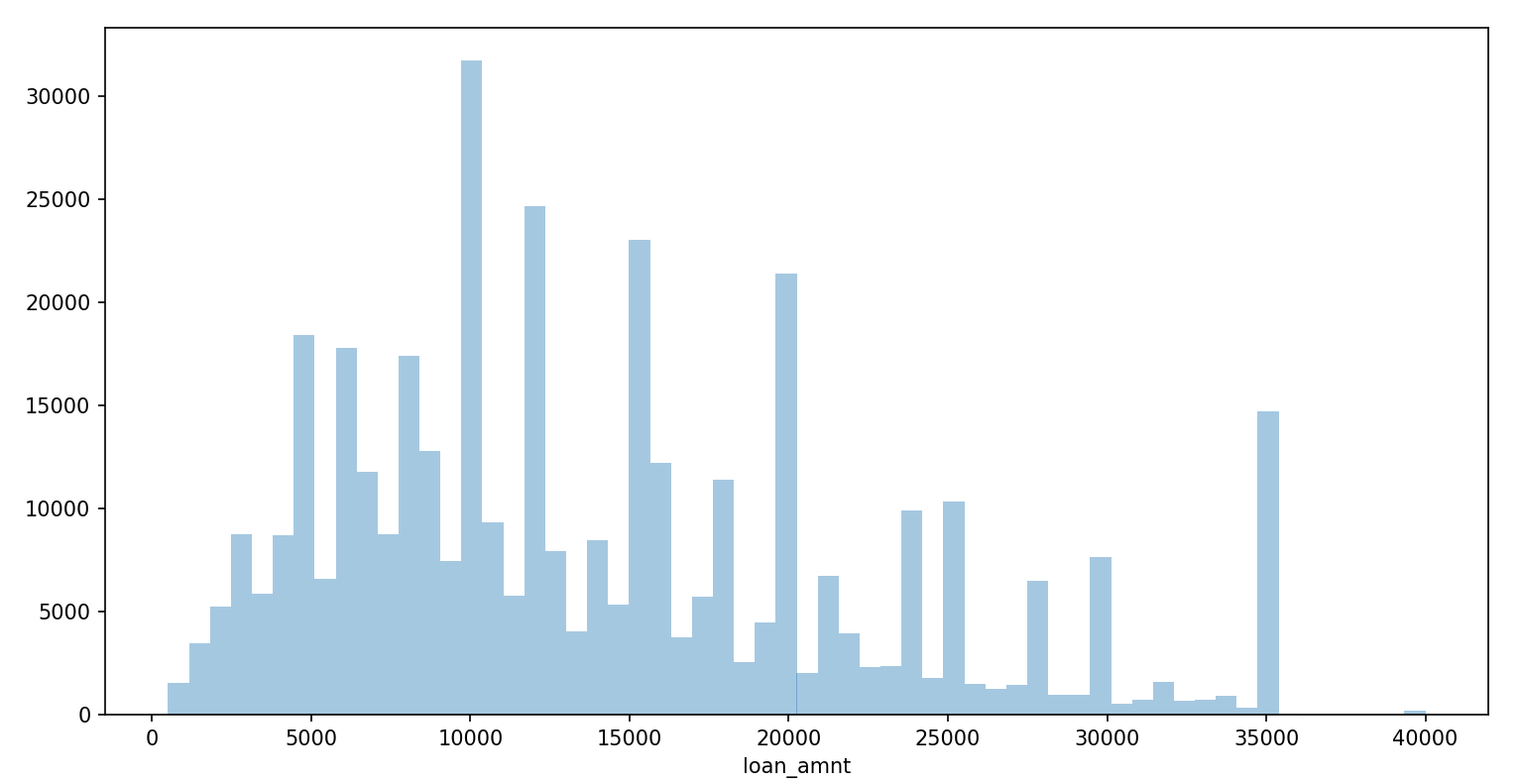


Figura 4 – Afisare histograma.

Calcularea corelației dintre toate variabilele numerice continue folosind metoda corr().

df.corr()

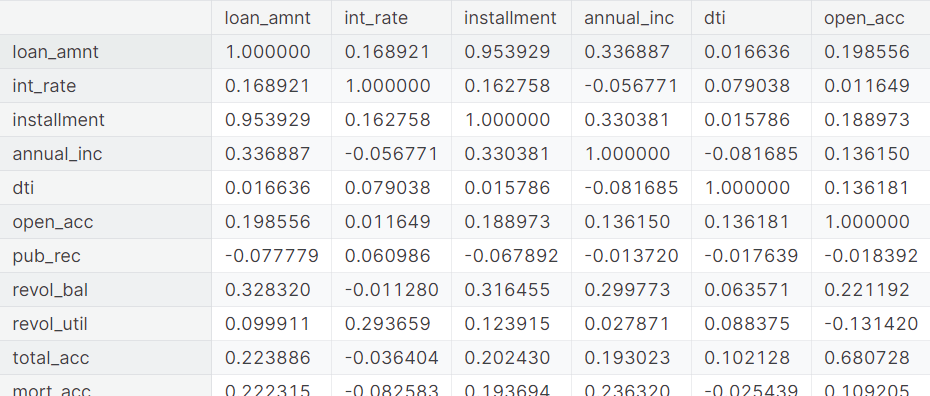


Figura 5 – Afișarea corelației

Vizualizarea corelației folosind heatmap.

**Heatmap**

def In12():  
 numeric\_columns = df.select\_dtypes(include=[np.number])  
 # Calculam matricea de corelatie  
 correlation\_matrix = numeric\_columns.corr()  
 # Afisam heatmap  
 plt.figure(figsize=(16, 6))  
 sns.heatmap(correlation\_matrix, annot=True)  
 plt.show()

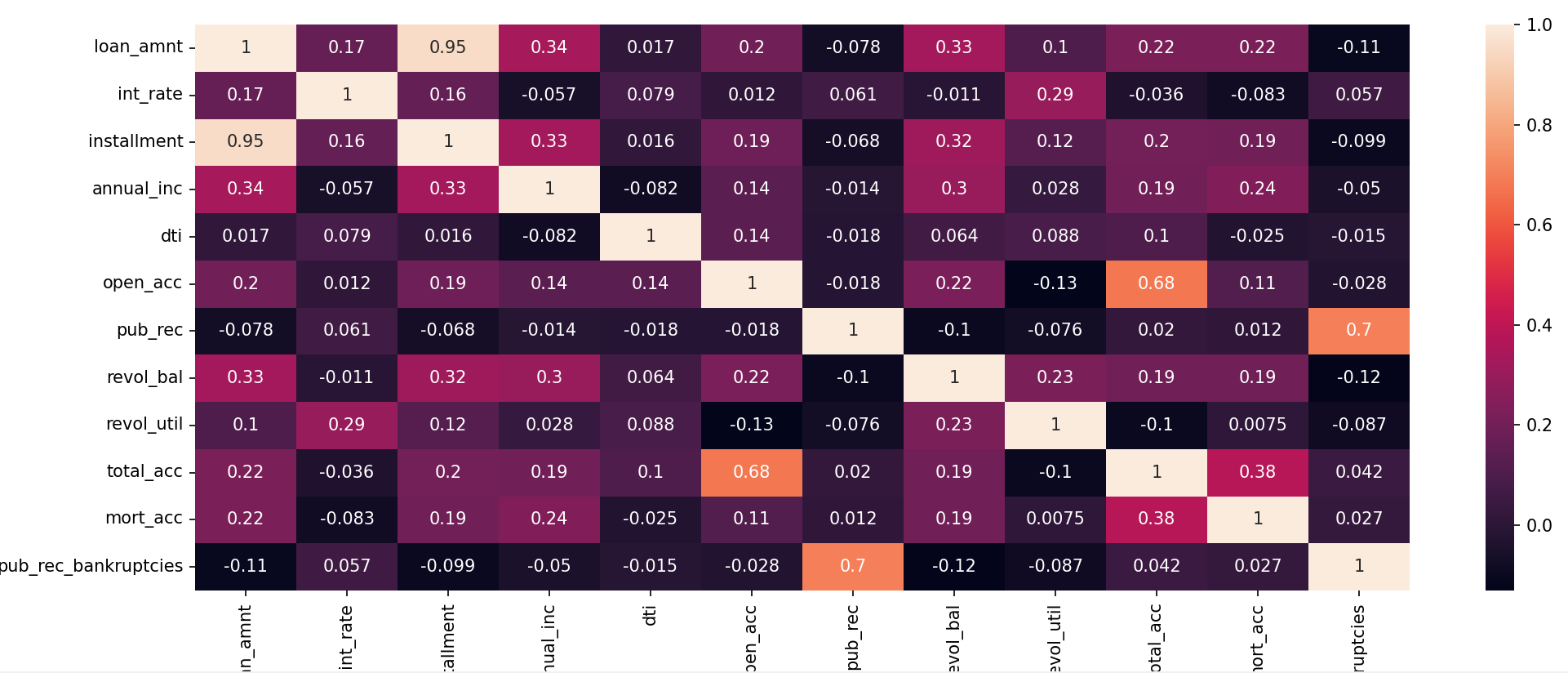


Figura 6 – Afisare heatmap

Observăm o corelație aproape perfectă.

Creare scatterplot

def In13():  
 #In13  
 sns.scatterplot(x="installment", y="loan\_amnt", data=df)  
 plt.show()

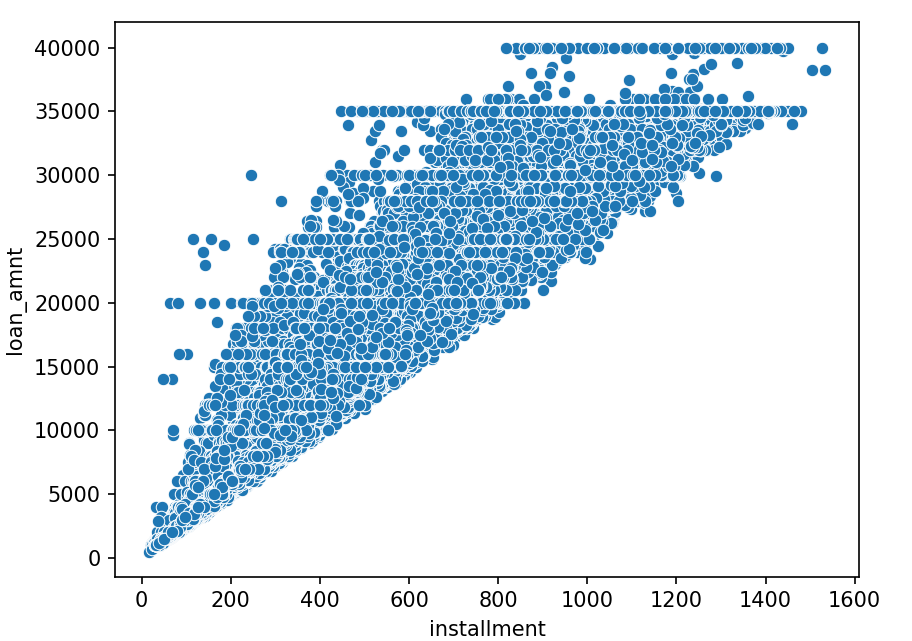


Figura 7 – Afisare scatterplot.

Creare unui boxplot care arată relația dintre loan\_status is loan\_amount.

def In14():  
 #In 14  
 sns.boxplot(x = "loan\_status", y = "loan\_amnt", data=df,)  
 plt.show()

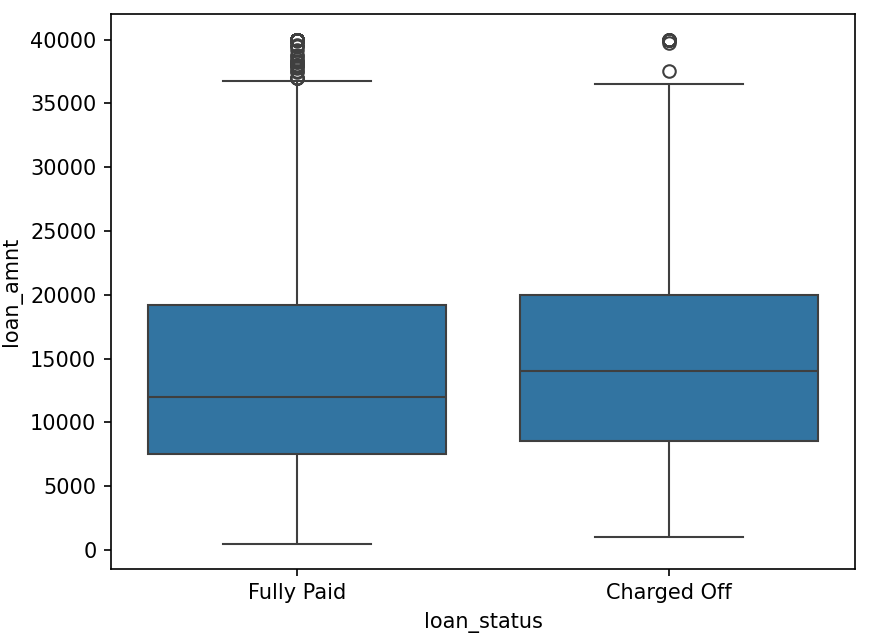


Figura 8 – Afisare boxplot.

Calcularea statisticei rezumate pentru loan\_amount, grupate după load\_status.

loan\_amount\_summary = df.groupby('loan\_status')['loan\_amnt'].describe()  
print(loan\_amount\_summary)

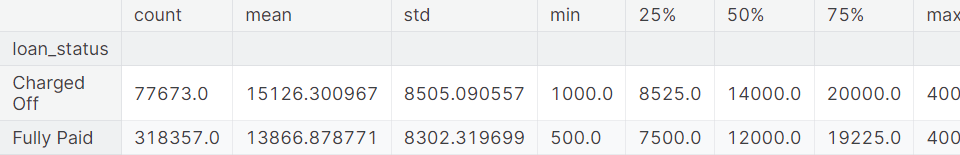


Figura 9 – Afisarea statisticii.

Afișarea notelor și subnotelor posibile unice.

unique\_grades = df['grade'].unique()  
print(unique\_grades)  
  
unique\_subgrades = df['sub\_grade'].unique()  
print(unique\_subgrades)

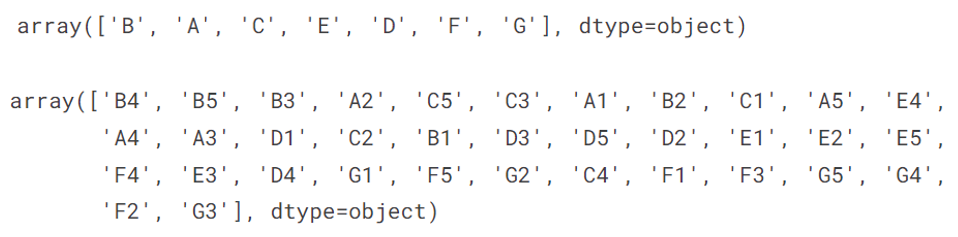


Figura 10 – Notele și subnotele

Crearea unui grafic de tip countplot pentru fiecare notă. Setăm hue cu valoarea loan\_status

def In18():  
 sns.countplot(x = "grade", hue="loan\_status", data=df)  
 plt.show()

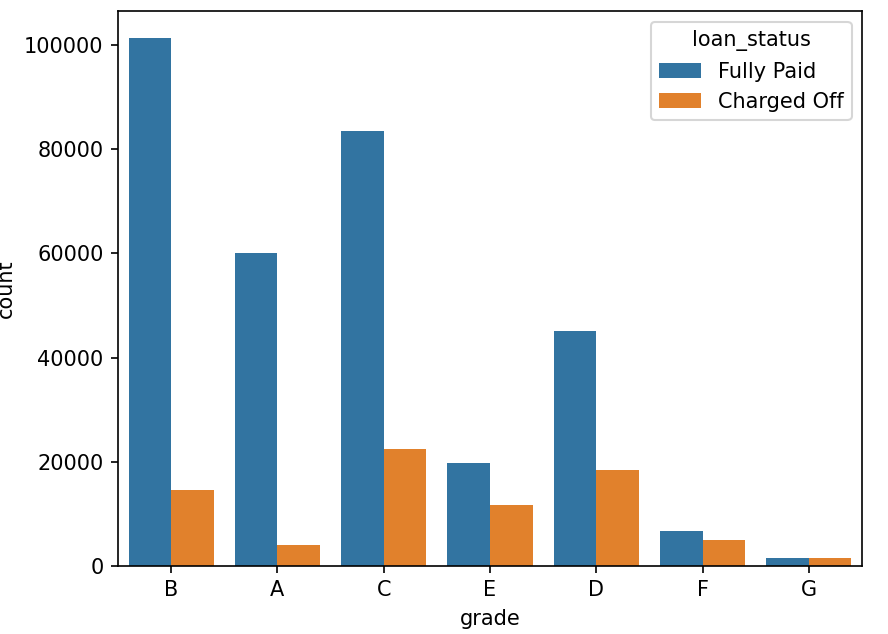


Figura 11 – Afișarea graficului de tip countplot

Afisarea count plot pentru fiecare subgrad. Explorăm atât toate împrumuturile făcute pe subclasare, cât și separate în funcție de loan\_status.

def In19():  
 plt.figure(figsize=(12, 5))  
 sns.countplot(x='sub\_grade', data=df, hue='loan\_status', order=sorted(df['sub\_grade'].unique()))  
 plt.show()

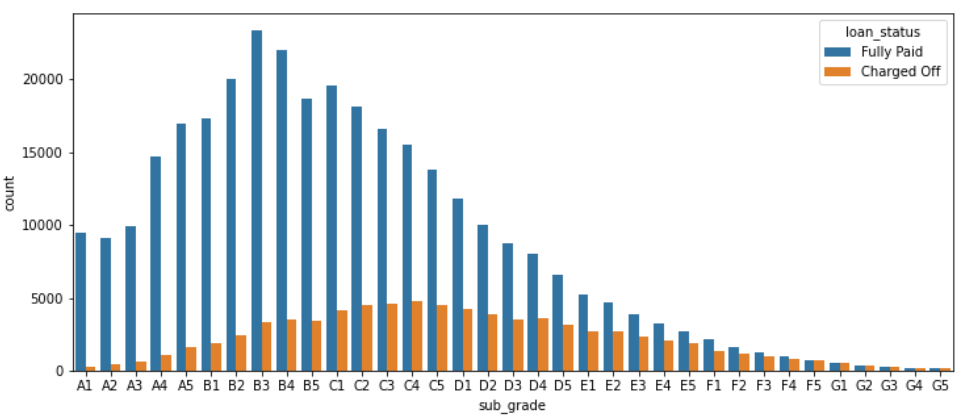


Figura 12 – Afisare countplot per fiecare subgrad

Subclasele F și G nu sunt plătite înapoi atât de des.

Sortăm și filtrăm datele pentru a vizualiza distribuția subnotelelor 'F' și 'G'.

def In20():  
 plt.figure(figsize=(12,4))  
 subgrade\_order = sorted(f\_and\_g['sub\_grade'].unique())  
 sns.countplot(x='sub\_grade',data=f\_and\_g,order = subgrade\_order,hue='loan\_status')  
 plt.show()

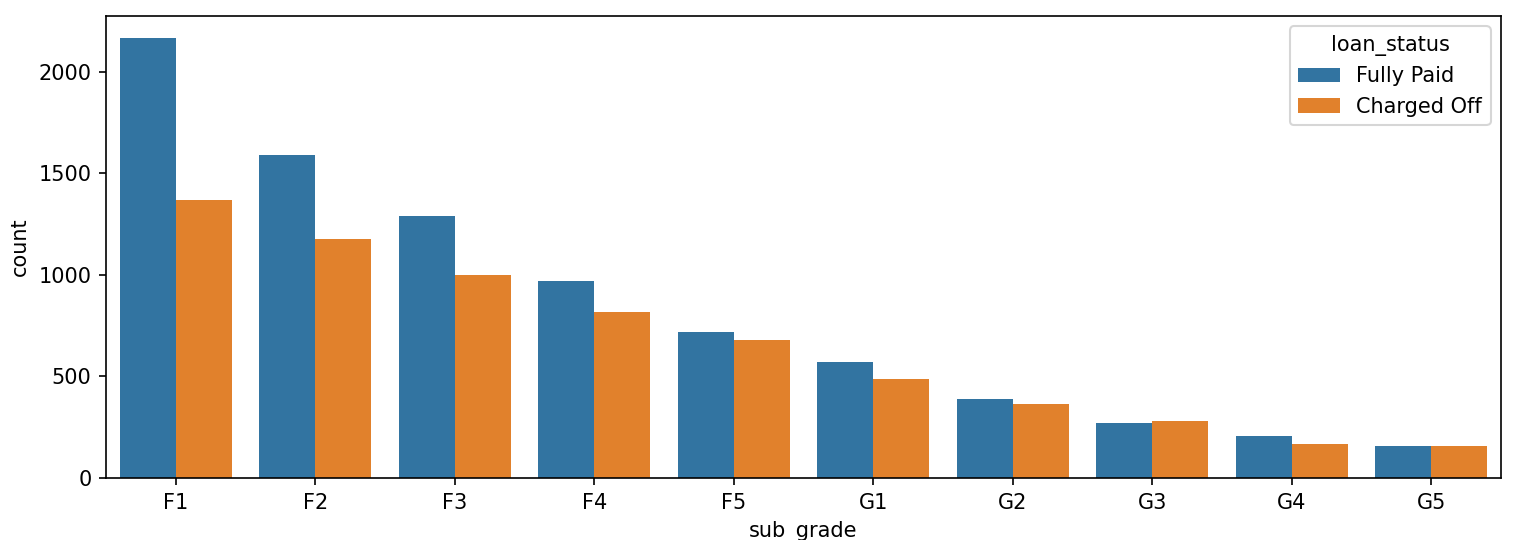


Figura 13 – Vizualizarea distributiei subnotelor F si G

Adaugăm o nouă coloană în DataFrame-ul original, numită 'loan\_repaid', care primește valoarea 1 pentru fiecare împrumut complet rambursat ('Fully Paid') și valoarea 0 dacă a fost Charged Off.

df['loan\_repaid'] = df['loan\_status'].apply(lambda x:1 if x=="Fully Paid" else 0

)

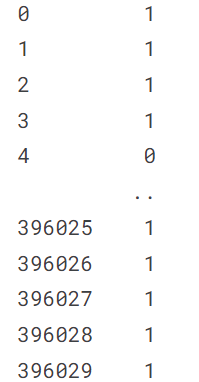


Figura 14 – Adaugare coloana noua loan\_repaid

Crearea unui grafic cu bare care să arate corelația caracteristicilor numerice cu noua coloană loan\_repaid.

def In23():  
 df.corr()['loan\_repaid'].sort\_values().drop('loan\_repaid').plot(kind='bar')  
 plt.show()

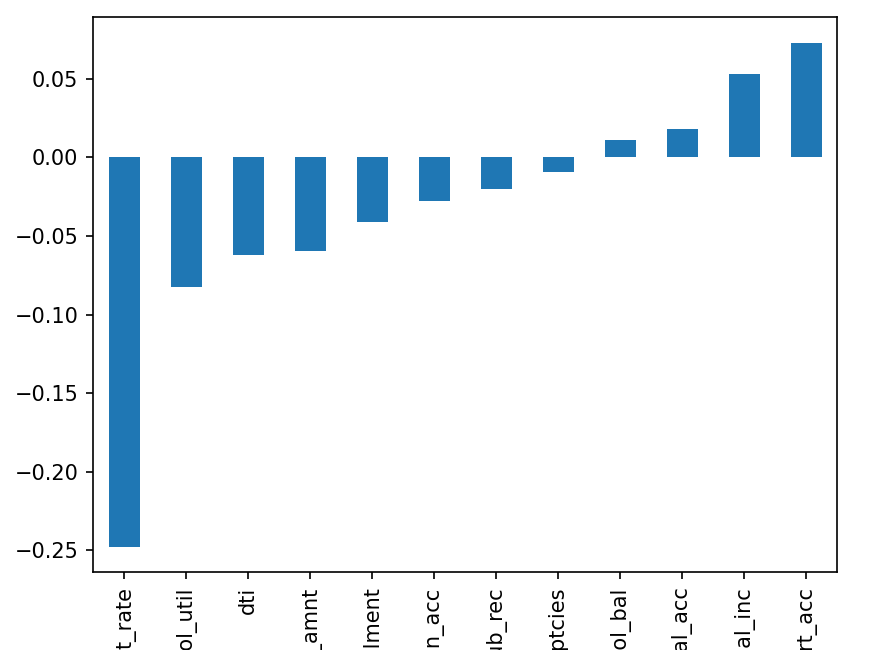


Figura 15 – Diagrama cu bare

**Secțiunea 2 : Preprocesarea datelor**

Eliminăm sau eliminăm datele lipsă. Eliminăm caracteristicile inutile sau repetitive. Convertim caracteristicile șirurilor categoriale în variabile fictive.

Calcularea numarului de valori pierdute per coloana.

df.isnull().sum()

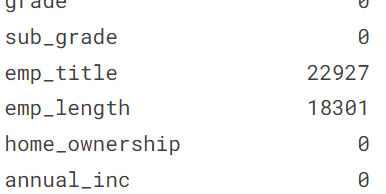


Figura 16 – Numarul de valori pierdute

**Date lipsă**

Explorarea datelor lipsă.

Afișarea lungimii dataframe-ului.



Figura 17 – Lungimea dataframe-ului

Convertăm în termeni de procent din cadrul total de date.

(df.isnull().sum()/len(df))\*100

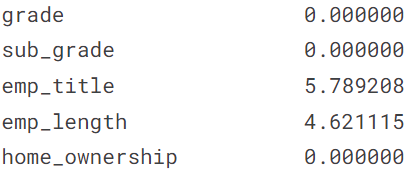


Figura 18 – Procentul de valori lipsa

Examinarea emp\_title și emp\_lenght pentru a vedea dacă va fi bine să le renunțăm.

feat\_info('emp\_title')  
  
feat\_info('emp\_length')



Figura 19 – Afisare informatii coloane

Aici se calculează și se afișează numărul de titluri de muncă unice din coloana 'emp\_title', apoi se afișează numărul de joburi pentru fiecare tip de titlu de muncă.

print(df['emp\_title'].nunique())

173105

print(df['emp\_title'].value\_counts())

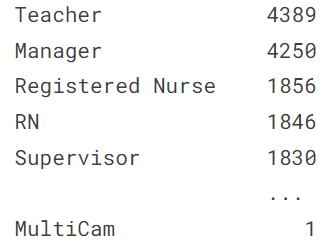


Figura 20 – Numărul de joburi pentru fiecare tip de titlu de muncă

Eliminam coloana emp\_title din DataFrame.

df = df.drop('emp\_title',axis=1)

print(df.columns)

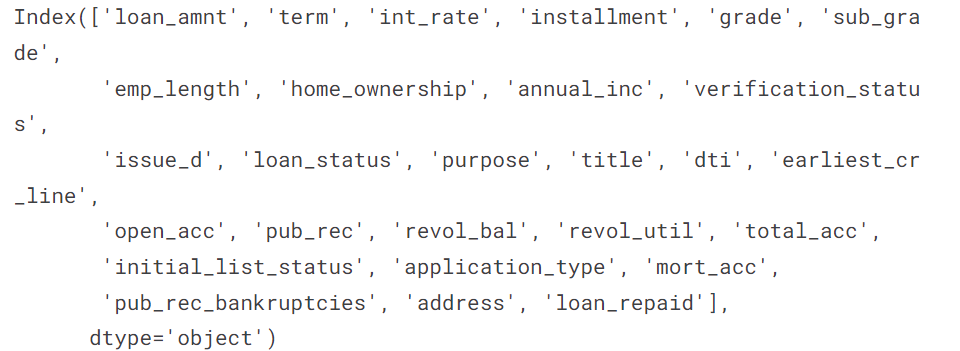


Figura 21 – Afișare coloane

Aici se creează un grafic de tip countplot pentru coloana 'emp\_length' din DataFrame, cu ordinea specificată pentru ani de experiență în muncă.

def In34():  
 plt.figure(figsize=(12,5))  
 sns.countplot(x='emp\_length',data=df, palette="hls", order=sorted(df['emp\_length'].dropna().unique()))  
 plt.show()

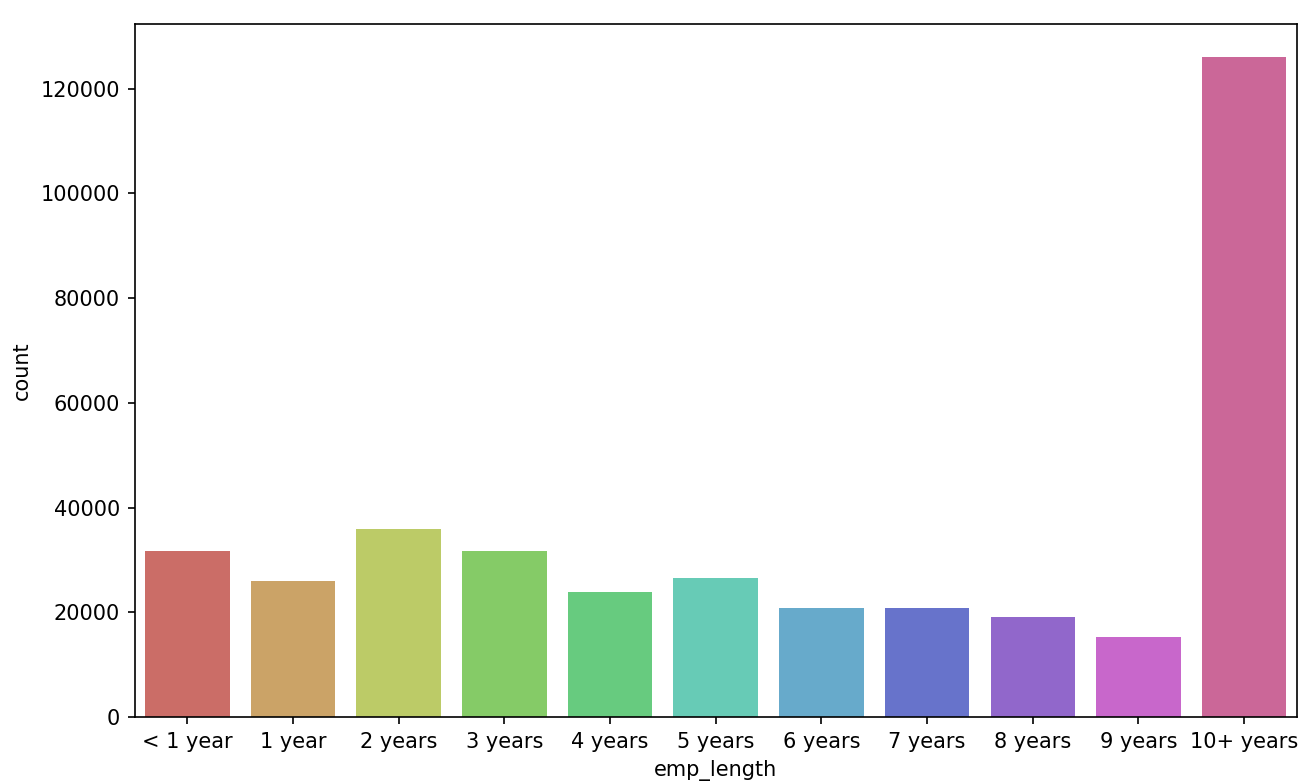


Figura 22 – Grafic countplot pentru coloana emp\_lenght

Creem un grafic de tip countplot pentru coloana 'emp\_length' din DataFrame-ul original, împărțit în funcție de starea împrumuturilor ('loan\_status'), cu ordinea specificată pentru ani de experiență în muncă.

def In35():  
 plt.figure(figsize=(12, 5))  
 sns.countplot(x='emp\_length', data=df, hue='loan\_status', order=sorted(df['emp\_length'].dropna().unique()))  
 plt.show()

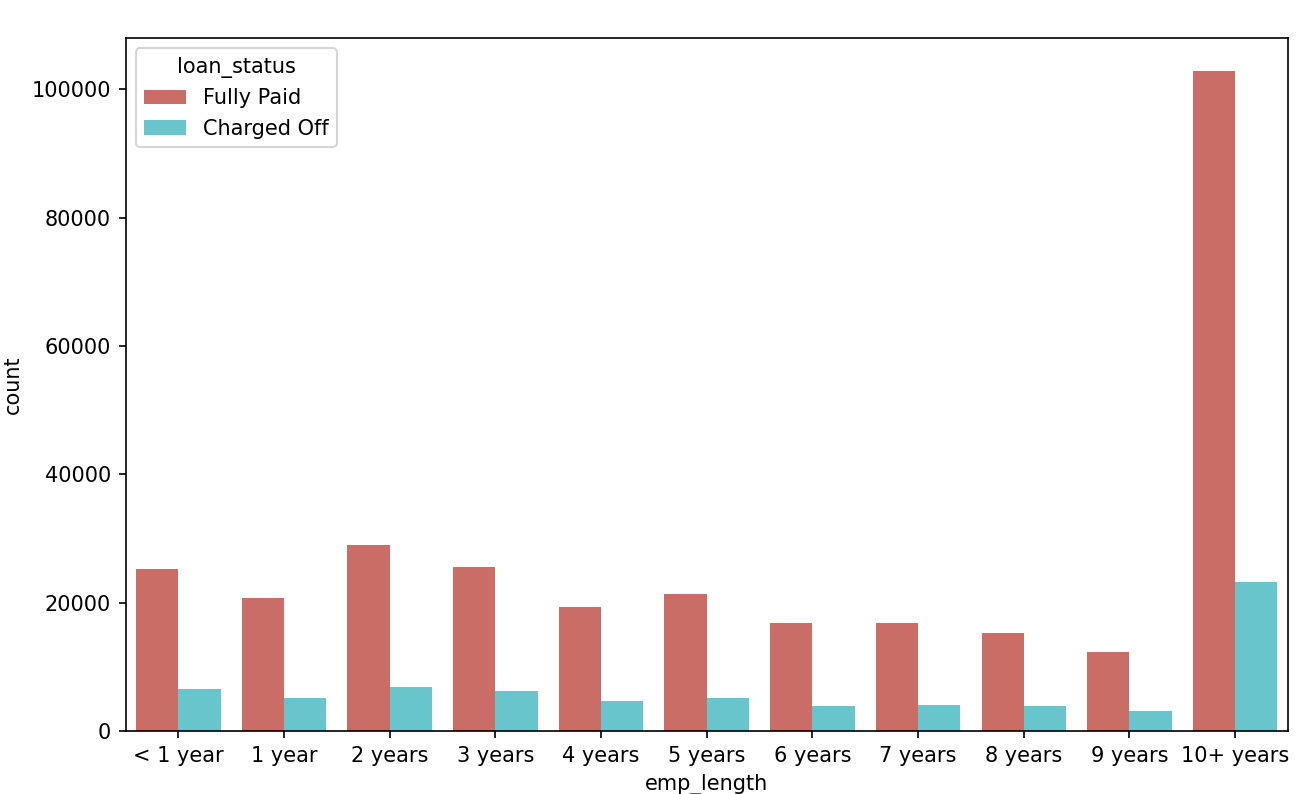


Figura 23 – Grafic countplot

Creem un nou DataFrame, numit 'charged\_off\_df', care conține doar rândurile în care starea împrumutului este 'Charged Off'. Apoi, se calculează și se afișează procentele de împrumuturi 'Charged Off' pentru fiecare categorie de ani de experiență în muncă din coloana 'emp\_length'. Apoi, se vizualizează aceste procente sub formă de diagramă bară.

def In36():  
 ChargedOff = df[df['loan\_status']=="Charged Off"].groupby("emp\_length").count()['loan\_status']  
 FullyPaid = df[df['loan\_status']=="Fully Paid"].groupby("emp\_length").count()['loan\_status']  
 emp\_len = ChargedOff/ FullyPaid  
 emp\_len.plot(kind='bar')  
 plt.show()

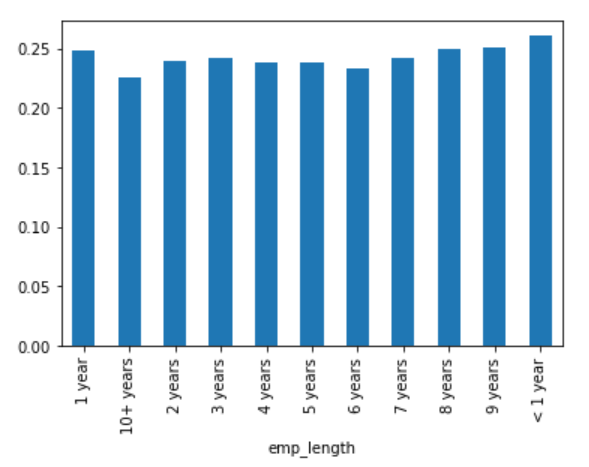


Figura 24 – Procentul de imprumuturi sub forma de diagrama

Ratele de taxare sunt extrem de similare pentru toate duratele de angajare. Eliminăm coloana emp\_lenght din DataFrame.

df = df.drop('emp\_length', axis=1)

print(df.columns)

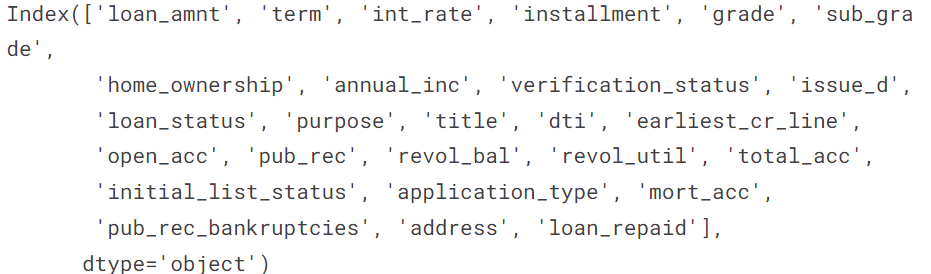


Figura 25 – Afișare coloane după eliminarea emp\_lenght

Revizuim DataFrame-ul pentru a vedea ce coloane de caracteristici au încă date lipsă.

Numărăm si afișăm titlurile de muncă din coloana title. Examinăm coloana de titlu față de coloana de scop.

print(df['purpose'].head(10))

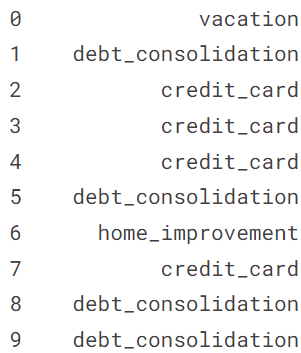


Figura 26 – Coloana purpose

print(df['title'].head(10))

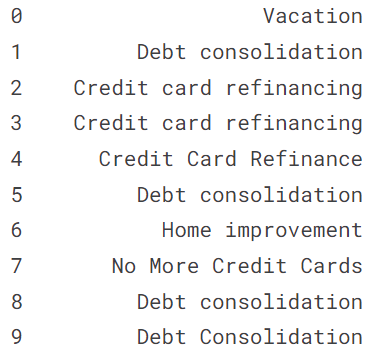


Figura 27 – Coloana title

Deoarece coloana de titlu este doar o simplă subcategorie, o eliminăm.

df = df.drop('title',axis=1)

print(df.columns)

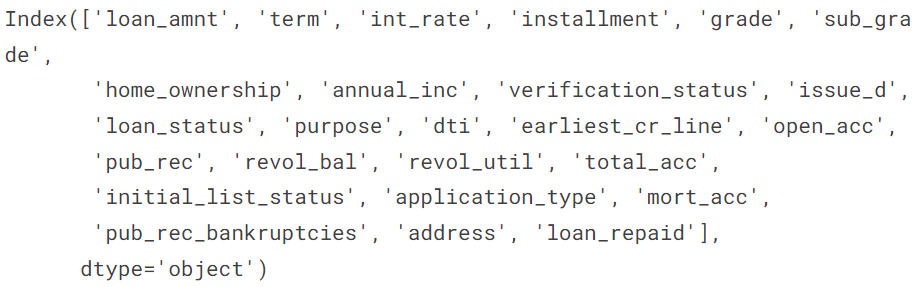


Figura 28 – Afișare coloane

Afișare ce reprezintă caracteristica mort\_acc.

feat\_info('mort\_acc')



Calculam numarul de valori din coloana mort\_acc.

df['mort\_acc'].value\_counts()

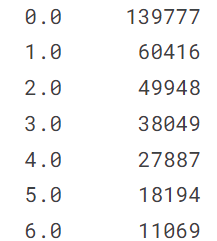


Figura 30 – Numarul de valori

Analizăm celelalte coloane pentru a vedea care se corelează cel mai mult cu mort\_acc.

df.corr()['mort\_acc'].sort\_values()

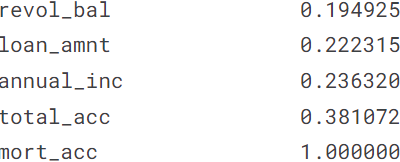


Figura 31 – Verificare care se corolează cu mort\_acc

Se observă că caracteristica total\_acc se corelează cu mort\_acc.

Grupăm cadrul de date după total\_acc și vom calcula valoarea medie pentru mort\_acc pe intrare total\_acc.

df.groupby('total\_acc').mean()

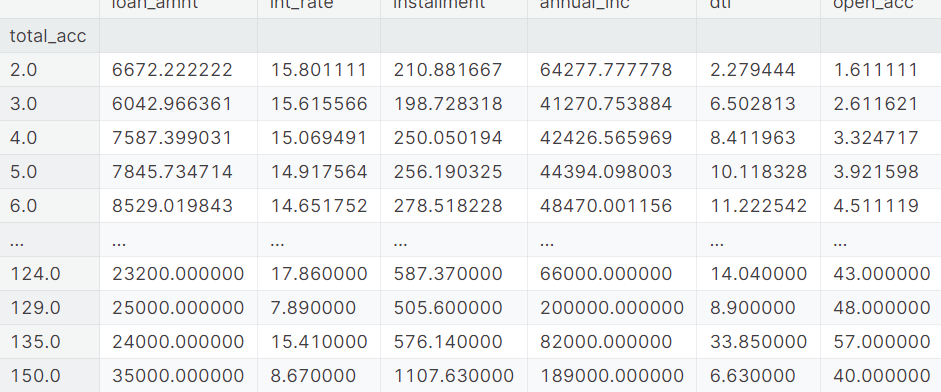


Figura 32 – Valoarea medie pentru mort\_acc pe intrare total\_acc

Completăm valorile mort\_acc care lipsesc pe baza valorii total\_acc. Dacă mort\_acc lipsește, atunci vom completa acea valoare lipsă cu valoarea medie corespunzătoare valorii sale total\_acc din seria creată. Aceasta implică utilizarea unei metode .apply() cu două coloane.

df['mort\_acc'] = df.apply(lambda x : total\_acc\_mean[x['total\_acc']] if (np.isnan(x['mort\_acc'])) else x['mort\_acc'],axis=1)

df.isnull().sum()

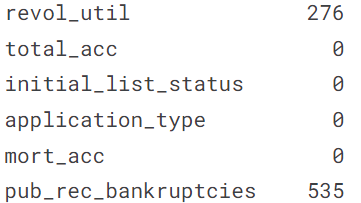


Figura 33 – Afișare valori nule

Revol\_util și pub\_rec\_bankruptcies au puncte de lipsă, dar reprezintă mai puțin de 0,5% din totalul datelor. Eliminăm rândurile care lipsesc datele din acele coloana, folosind dropna().

df = df.dropna()  
df.isnull().sum()

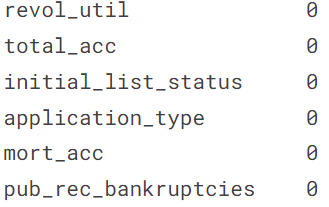


Figura 34 – Afișare date nule

**Variabile categoriale și variabile fictiv**

df.select\_dtypes(['object']).columns

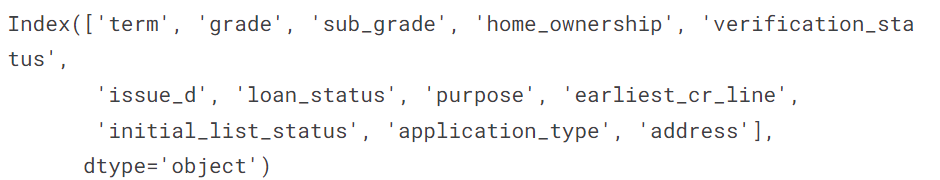
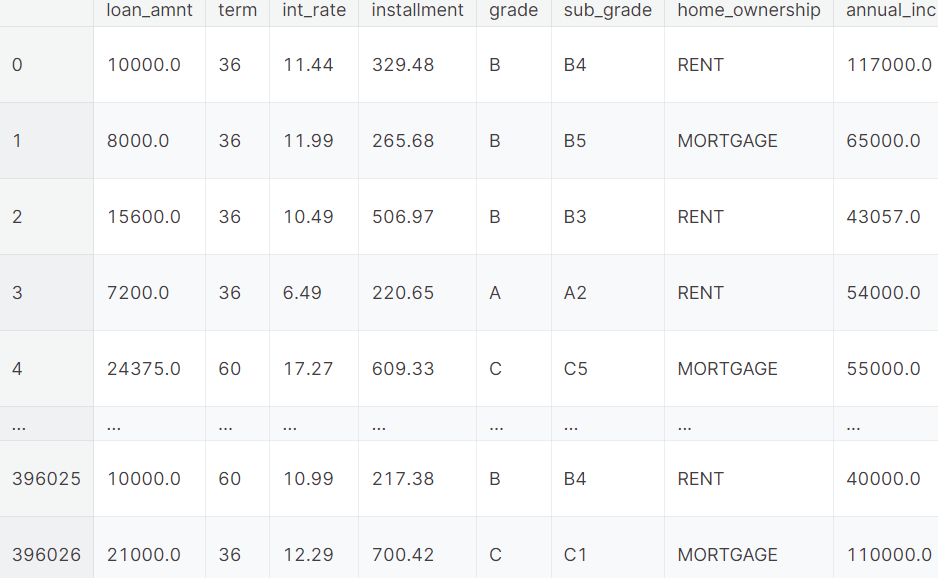


Figura 35 – Coloanele care sunt în prezent nonnumerice

Convertirea termenului caracteristică, într-un tip de date numerice cu 36 sau 60 de numere întregi folosind .apply() sau map().

df['term'] = df['term'].apply(lambda term: int(term[:3]))

print(df)



Deoarece grade face parte din sub\_grade, renunțăm la funcția dată.

df =df.drop('grade',axis =1)

Convertim sub\_grade în variabile dummy. Apoi concatinăm aceste coloane noi la cadrul de date original. Renunțăm la coloana inițială de sub\_grade și adăugăm drop\_first = True la apelul get\_dummies.

dummies = pd.get\_dummies(df['sub\_grade'],drop\_first=True)  
print(dummies)

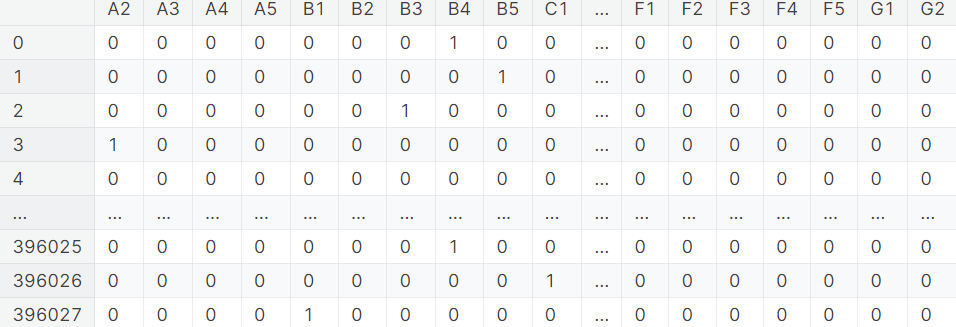


Figura 37 – Afișare valori fictive

df = pd.concat([df.drop('sub\_grade',axis=1),dummies],axis=1)  
print(df.columns)

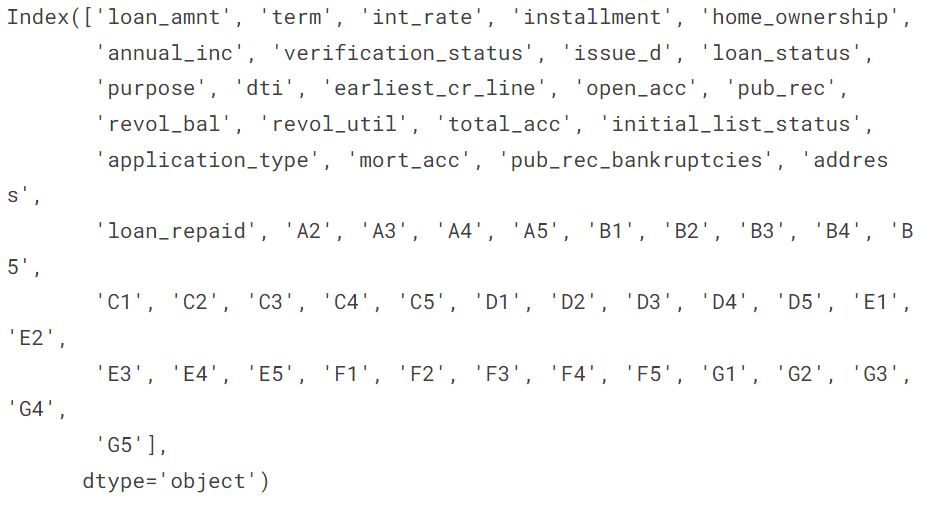


Figura 38 – Coloanele curente

Convertim coloanele verification\_status, application\_type, initial\_list\_status, purpose în variabile fictive și le concatinăm cu cadrul de date original. Setăm drop\_first = True și să aruncăm coloanele originale.

dummies = pd.get\_dummies(df[['verification\_status', 'application\_type','initial\_list\_status','purpose' ]],drop\_first=True)  
df = df.drop(['verification\_status', 'application\_type','initial\_list\_status','purpose'],axis=1)  
df = pd.concat([df,dummies],axis=1)

Examinăm value\_counts pentru coloana home\_ownership.

df['home\_ownership'].value\_counts()

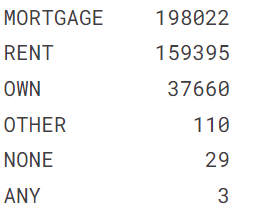


Figura 39 – Detalii colloana home\_ownership

Convertim în variabile fictive, dar înlocuim NONE și ANY cu OTHER, astfel ca să avem doar 4 categorii MORTAGE,RENT,OWN,OTHER. Apoi le concatinăm cu cadrul de date original. Nu uităm să setăm drop\_first=True și aruncăm coloanele originale.

df['home\_ownership']=df['home\_ownership'].replace(['NONE', 'ANY'], 'OTHER')  
  
dummies = pd.get\_dummies(df['home\_ownership'],drop\_first=True)  
df = df.drop('home\_ownership',axis=1)  
df = pd.concat([df,dummies],axis=1)  
df.columns

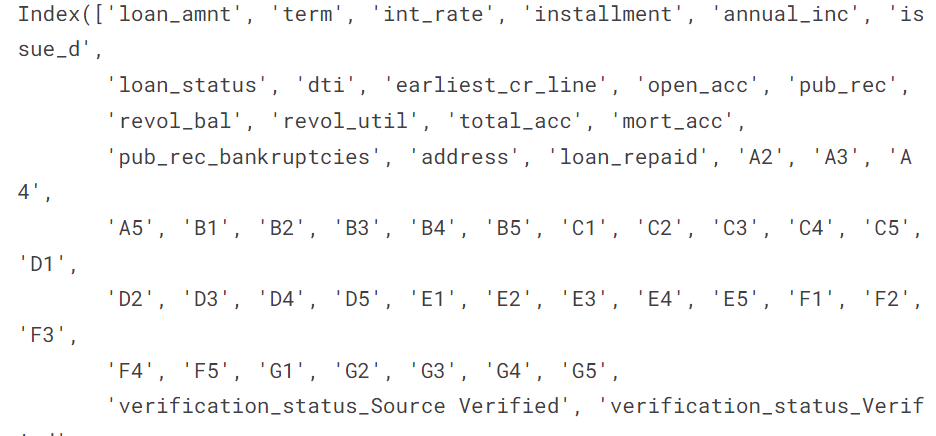


Figura 40 – Afișare rezultat

**Adresa**

Creăm o coloană nouă zip\_code care extrage codul poștal din coloana cu adresă.

df['zip\_code'] = df['address'].apply(lambda x:x.split(' ')[-1])  
df['zip\_code'].value\_counts()

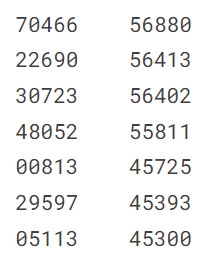


Figura 41 – Afișare numărului per fiecare zip\_code

Transformăm coloana zip\_code în variabile fictive folosind panda. Concatinăm rezultatul șu aruncăm coloana cu codul poștal înpreună cu eliminarea coloanei cu adresă.

dummies = pd.get\_dummies(df['zip\_code'],drop\_first=True)  
df = df.drop(['zip\_code','address'],axis=1)  
df = pd.concat([df,dummies],axis=1)

**Issues\_d**

Nu știm dacă dacă în prealabil se va acorda sau nu un împrumut atunci când folosim modelul nostru, respectiv nu avem un issue\_date și ștergem coloana.

df = df.drop('issue\_d',axis=1)

**Train Test Slit**

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

Ștergem coloanele load\_status pe care am creat-o anterior, deoarece este un dublicat al coloanei loan\_repaid. Vom folosi coloana loan\_repaid, deoarece este deja în 0 și 1.

df = df.drop('loan\_status',axis=1)  
print(df.columns)

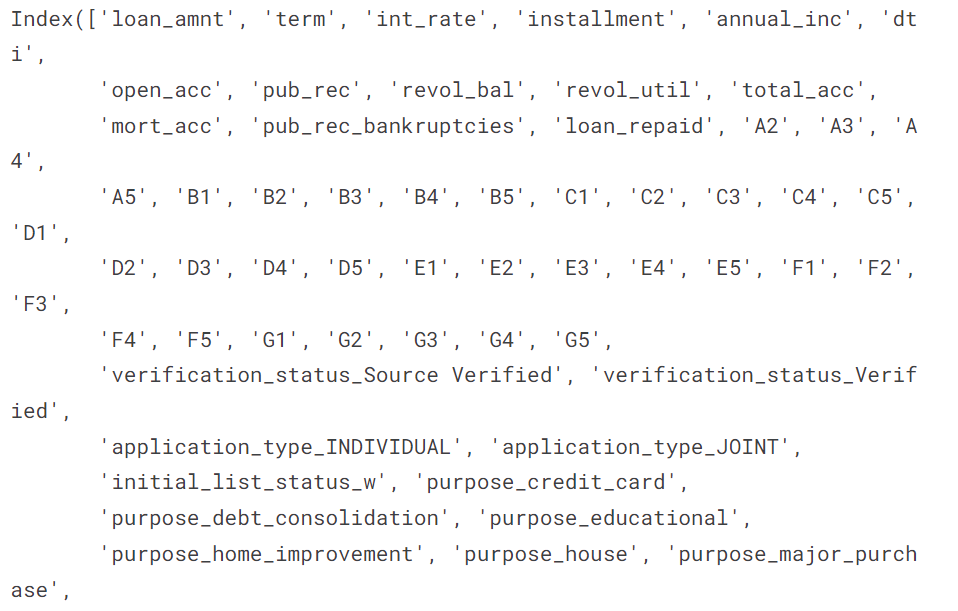


Figura 42 – Afișare coloane

Setăm variabilele x și y la valorile caracteristicelor și etichetei.

X= df.drop('loan\_repaid',axis=1).values  
y= df['loan\_repaid'].values

**Opțional**

Folosim sample() pentru a prelua o mostră din cele peste 490 000 de intrări pentru a economisi timp la antrenament.

sample = df.sample(frac=0.1,random\_state=101)  
print(len(df))



Efectuăm o împărțire train/test cu test\_size = 0,2 și o stare aleatorie de 101.

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

**Normalizind the Data**

Utilizăm un MinMaxScaler pentru a normaliza datele caracteristicilor X\_train și X\_test.

from sklearn.preprocessing import MinMaxScaler  
scaler = MinMaxScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_test = scaler.transform(X\_test)

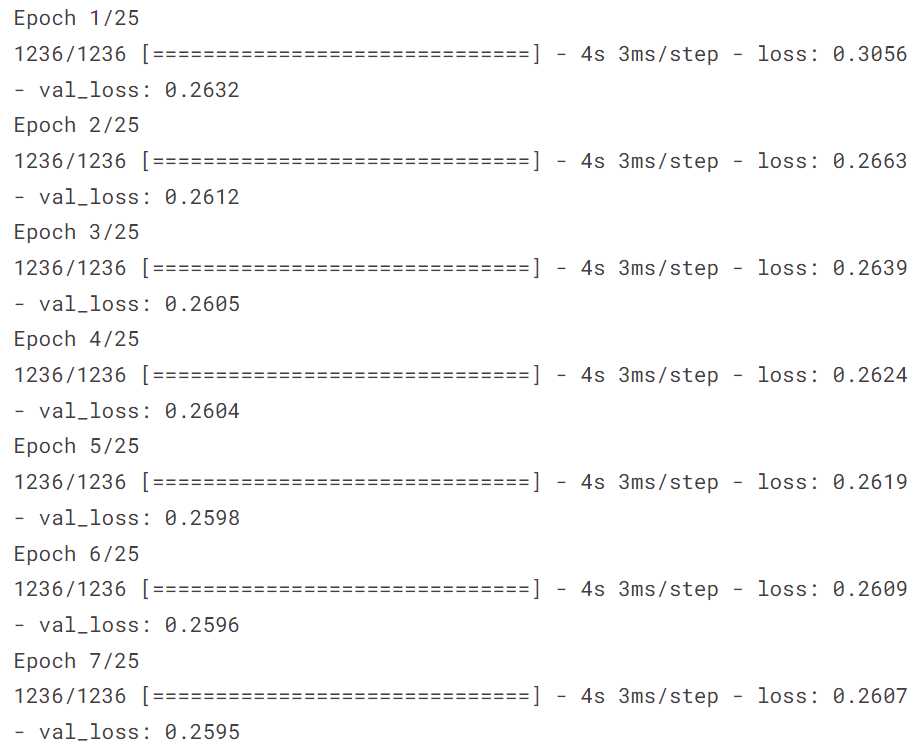
**Crearea modelului**

Construim un model secvențial pentru a fi instruit cu privire la date. Vom folosi un model care merge 78-39-19-1 neuron de ieșire.

from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense,Dropout  
  
model = Sequential()  
model.add(Dense(78, activation='relu'))  
model.add(Dropout(0.2))  
model.add(Dense(39, activation='relu'))  
model.add(Dropout(0.2))  
model.add(Dense(19, activation='relu'))  
model.add(Dropout(0.2))  
model.add(Dense(units=1,activation='sigmoid'))  
model.compile(loss='binary\_crossentropy', optimizer='adam')

Potrivim modelul la datele de antrenament pentru cel puțin 25 de epoci. De asemenea, adăugăm datele de validare pentru trasarea ulterioară. Opțional adăugăm o dimensiune batch\_size de 256.

model.fit(x=X\_train,   
 y=y\_train,   
 epochs=25,  
 batch\_size=256,  
 validation\_data=(X\_test, y\_test),   
 )



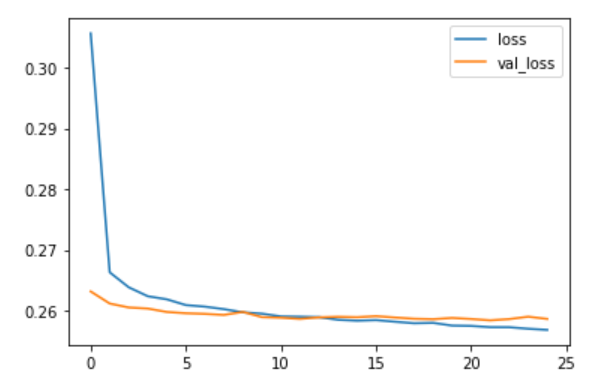
**Salvarea modelului**

from tensorflow.keras.models import load\_model  
model.save('keras.h5')

**Secțiunea 3 : Evaluare performanței modelului**

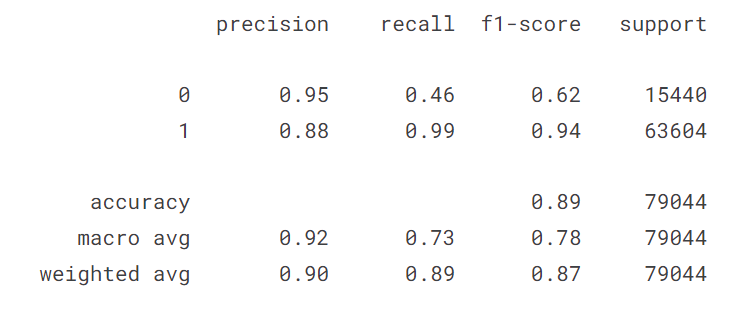
Înregistrăm pierderea de validare față de pierderea de antrenamente.

loss = pd.DataFrame(model.history.history)  
loss[['loss','val\_loss']].plot()

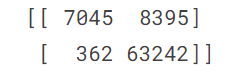


Creare predicție din setul X\_test și afișăm un raport de clasificare și o matrice de confuzie pentru setul X\_test.

predictions = model.predict\_classes(X\_test)  
from sklearn.metrics import classification\_report,confusion\_matrix  
print(classification\_report(y\_test,predictions))

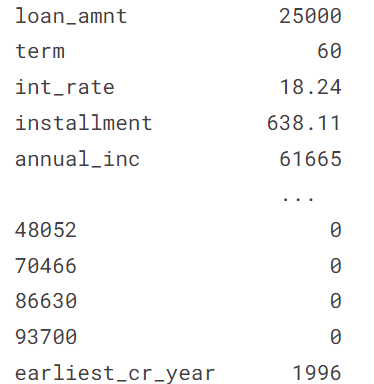


print(confusion\_matrix(y\_test,predictions))

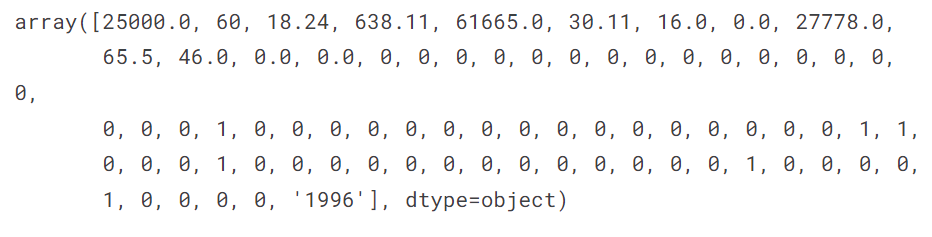


Testare

import random  
random.seed(101)  
random\_ind = random.randint(0,len(df))  
  
new\_customer = df.drop('loan\_repaid',axis=1).iloc[random\_ind]  
new\_customer



new\_customer.values



**Verificare persoana**

df.iloc[random\_ind]['loan\_repaid']



**Concluzie**

Lucrarea începe cu importarea și explorarea datelor dintr-un fișier CSV. Sunt efectuate analize exploratorii, inclusiv vizualizarea datelor utilizând diverse tipuri de grafice și calcularea corelațiilor. Se efectuează preprocesarea datelor, inclusiv gestionarea valorilor lipsă și transformarea caracteristicilor categorice în variabile fictive. Se construiește și se antrenează un model de învățare automată folosind un MinMaxScaler și un model secvențial de rețea neuronală. Performanța modelului este evaluată utilizând metrici precum pierderea și raportul de clasificare, iar modelul este salvat pentru utilizări ulterioare.