Welcome to my JUPYTER

Flow of Content:

1 Introduction

- Info's about datasets

2. Librarys

- Importing Librarys
- Importing Dataset

3. Knowning the data

- 3.1 Looking the Type of Data
- 3.2 Shape of data
- 3.3 Null Numbers
- 3.4 Unique values
- 3.5 The first rows of our dataset

4. Exploring some Variables

- 4.1 Ploting some graphical and descriptive informations

5. Correlation of data

- 5.1 Correlation Data

6. Preprocess

- 6.1 Importing Librarys
- 6.2 Setting X and Y
- 6.3 Spliting the X and Y in train and test

7. Model

Logistic Regression

- Score values
- Cross Validation
- ROC Curve

1 Introduction

Context

```
The original dataset contains 1000 entries with 20 categorial/symbolic attributes prepared by Prof. Hofmann. In this dataset, each entry represents a person
who takes a credit by a bank. Each person is classified as good or bad credit risks according to the set of attributes. The link to the original dataset can be
found below.
```

Content

```
It is almost impossible to understand the original dataset due to its complicated system of categories and symbols. Thus, I wrote a small Python script to
convert it into a readable CSV file. Several columns are simply ignored, because in my opinion either they are not important or their descriptions are obscure.
The selected attributes are:
```

Age (numeric)

Sex (text: male, female)

Job (numeric: 0 - unskilled and non-resident, 1 - unskilled and resident, 2 - skilled, 3 - highly skilled)

Housing (text: own, rent, or free)

Saving accounts (text - little, moderate, quite rich, rich)

Checking account (numeric, in DM - Deutsch Mark)

Credit amount (numeric, in DM)

Duration (numeric, in month)

Purpose(text: car, furniture/equipment, radio/TV, domestic appliances, repairs, education, business, vacation/others

Risk (Value target - Good or Bad Risk)

2.Librarys:

Importing Librarys

Importing Dataset

In [222…
```
import pandas as pd,numpy as np,os
import matplotlib.pyplot as plt
import seaborn as sns
```

In [223…
```
os.chdir('D:\machine learning\Raw data')
```

In [224…
```
df_credit=pd.read_csv('german_credit_data.csv')
```

3. Knowning the data

3.1 Looking the Type of Data

3.2 Shape of data

3.3 Null Numbers

3.4 Unique values

3.5 The first rows of our dataset

In [225…
```
df_credit.isnull().sum()
```

Out[225…
```
Unnamed: 0            0
Age                   0
Sex                   0
Job                   0
Housing               0
Saving accounts     183
Checking_account    394
Credit_amount         0
Duration              0
Purpose               0
Risk                  0
dtype: int64
```

In [226…
```
df_credit.isnull().mean()*100
```

Out[226…
```
Unnamed: 0           0.0
Age                  0.0
Sex                  0.0
Job                  0.0
Housing              0.0
Saving accounts     18.3
Checking_account    39.4
Credit_amount        0.0
Duration             0.0
Purpose              0.0
Risk                 0.0
dtype: float64
```

In [227…
```
df_credit.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        1000 non-null   int64
 1   Age               1000 non-null   int64
 2   Sex               1000 non-null   object
 3   Job               1000 non-null   int64
 4   Housing           1000 non-null   object
 5   Saving accounts   817 non-null    object
 6   Checking_account  606 non-null    object
 7   Credit_amount     1000 non-null   int64
 8   Duration          1000 non-null   int64
 9   Purpose           1000 non-null   object
 10  Risk              1000 non-null   object
dtypes: int64(5), object(6)
memory usage: 86.1+ KB
```

In [228…
```
df_credit.nunique()
```

Out[228…
```
Unnamed: 0          1000
Age                   53
Sex                    2
Job                    4
Housing                3
Saving accounts        4
Checking_account       3
Credit_amount        921
Duration              33
Purpose                8
Risk                   2
dtype: int64
```

In [229…
```
df_credit.head()
```

Out[229…

	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking_account	Credit_amount	Duration	Purpose	Risk
0	0	67	male	2	own	NaN	little	1169	6	radio/TV	good
1	1	22	female	2	own	little	moderate	5951	48	radio/TV	bad
2	2	49	male	1	own	little	NaN	2096	12	education	good
3	3	45	male	2	free	little	little	7882	42	furniture/equipment	good
4	4	53	male	2	free	little	little	4870	24	car	bad

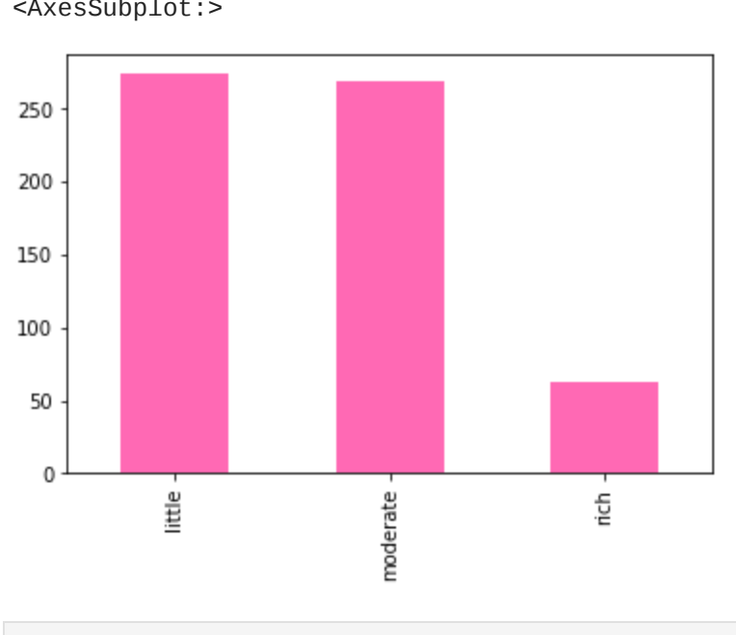
In [230…
```
df_credit.shape
```

Out[230… (1000, 11)

null value

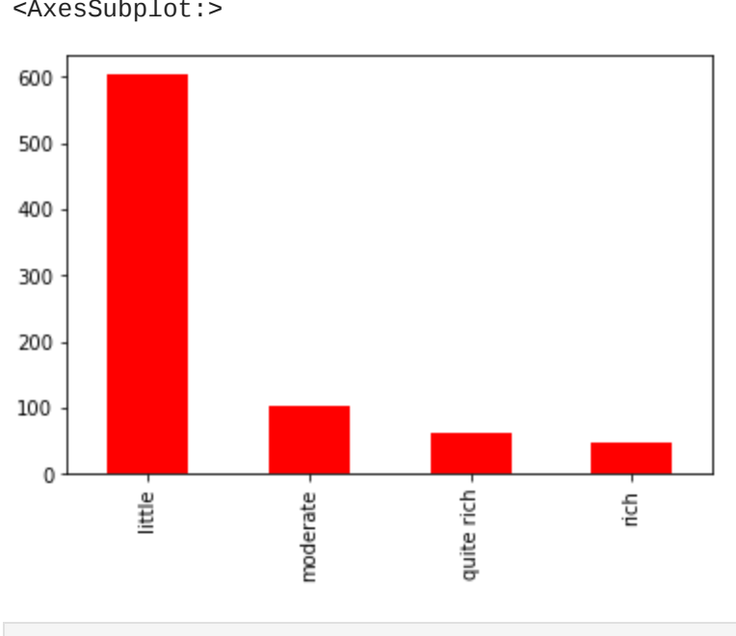
In [231…
```
df_credit['Checking_account'].value_counts().plot.bar(color='hotpink')
```

Out[231… <AxesSubplot:>



In [232…
```
df_credit['Saving accounts'].value_counts().plot.bar(color='r')
```

Out[232… <AxesSubplot:>



In [233…
```
def impute_nan(df,variable):
    most_frequent_category=df[variable].mode()[0]
    df[variable].fillna(most_frequent_category,inplace=True)
```

In [234…
```
for feature in ['Checking_account','Saving accounts']:
    impute_nan(df_credit,feature)
```

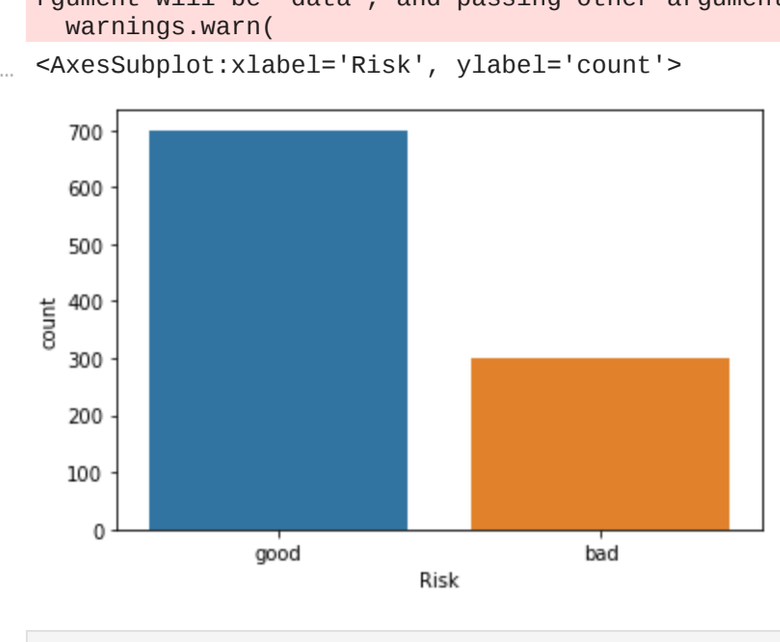
In [235…
```
df_credit.isnull().sum()
```

Out[235…
```
Unnamed: 0          0
Age                 0
Sex                 0
Job                 0
Housing             0
Saving accounts     0
Checking_account    0
Credit_amount       0
Duration            0
Purpose             0
Risk                0
dtype: int64
```

In [236…
```
sns.countplot('Risk', data=df_credit)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

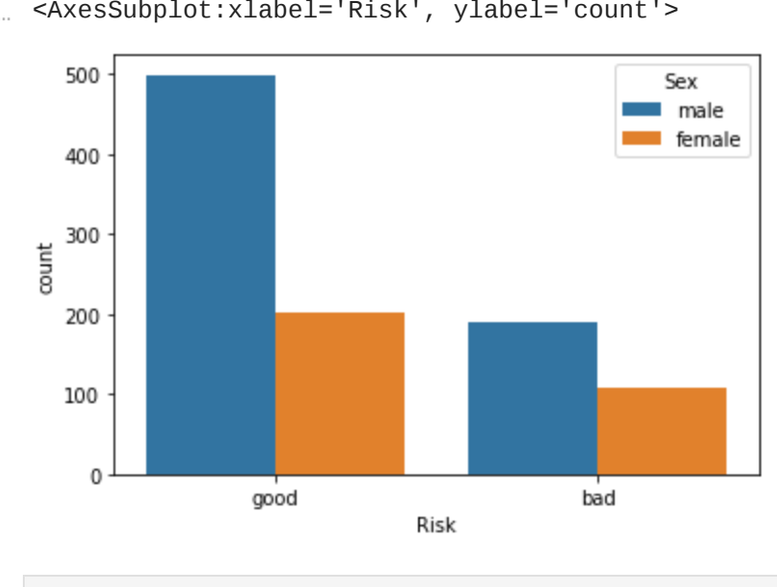
Out[236… <AxesSubplot:xlabel='Risk', ylabel='count'>



In [237…
```
sns.countplot('Risk', hue='Sex',data=df_credit)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

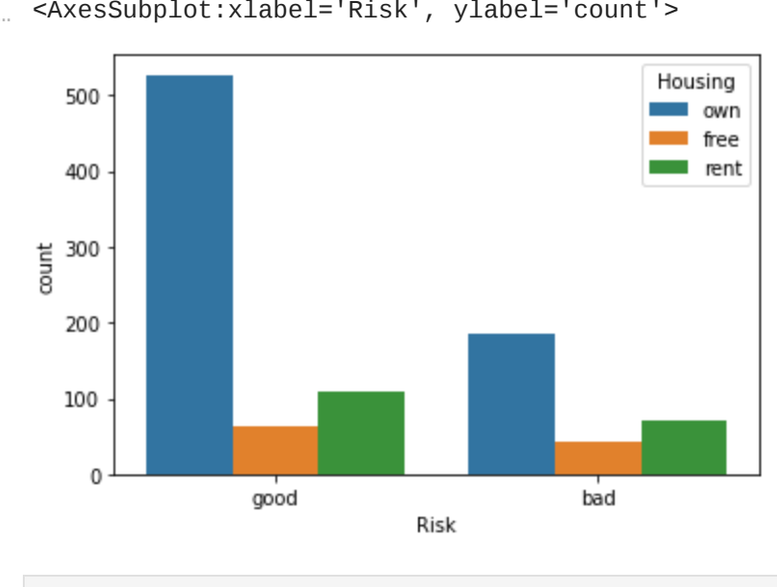
Out[237… <AxesSubplot:xlabel='Risk', ylabel='count'>



In [238…
```
sns.countplot('Risk', hue='Housing',data=df_credit )
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

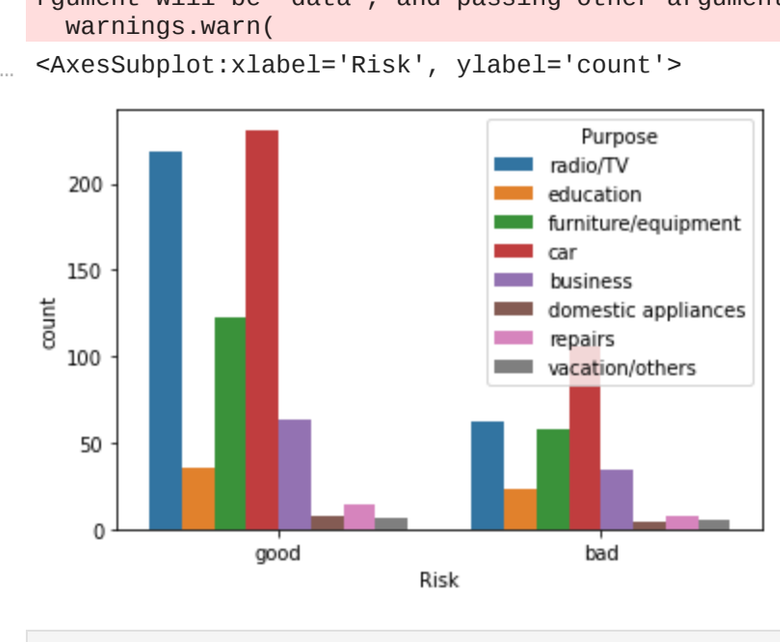
Out[238… <AxesSubplot:xlabel='Risk', ylabel='count'>



In [239…
```
sns.countplot('Risk', hue='Purpose',data=df_credit )
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[239… <AxesSubplot:xlabel='Risk', ylabel='count'>



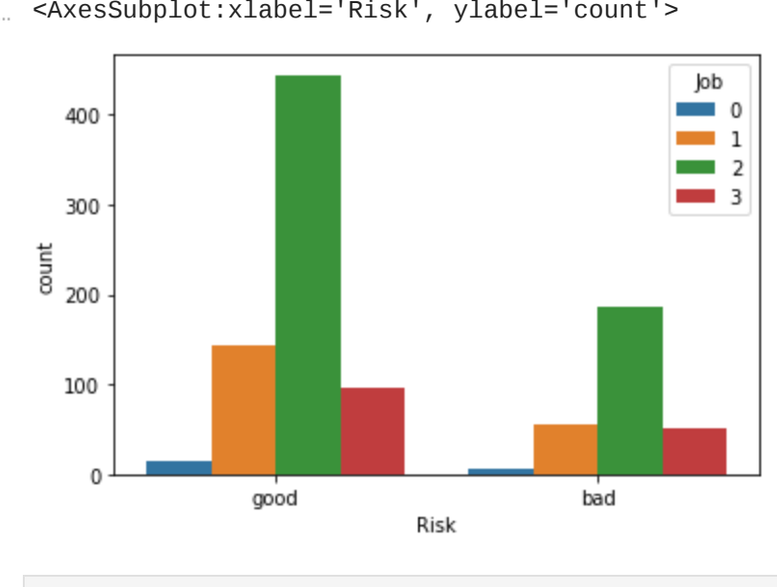
In [240…
```
df_credit.columns
```

Out[240…
```
Index(['Unnamed: 0', 'Age', 'Sex', 'Job', 'Housing', 'Saving accounts',
       'Checking_account', 'Credit_amount', 'Duration', 'Purpose', 'Risk'],
      dtype='object')
```

In [241…
```
sns.countplot('Risk', hue='Job',data=df_credit )
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

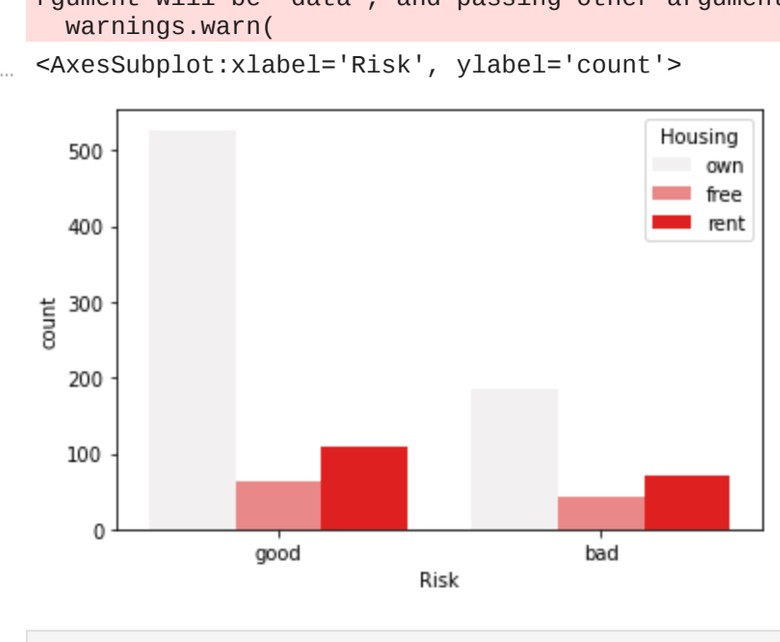
Out[241… <AxesSubplot:xlabel='Risk', ylabel='count'>



In [242…
```
sns.countplot("Risk",hue='Housing',data=df_credit,color='r')
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[242… <AxesSubplot:xlabel='Risk', ylabel='count'>



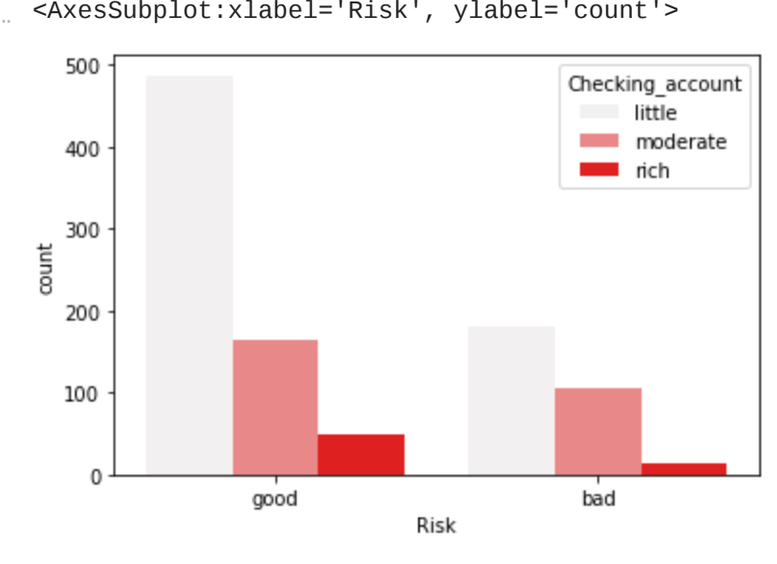
In [243…
```
df_credit.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        1000 non-null   int64
 1   Age               1000 non-null   int64
 2   Sex               1000 non-null   object
 3   Job               1000 non-null   int64
 4   Housing           1000 non-null   object
 5   Saving accounts   1000 non-null   object
 6   Checking_account  1000 non-null   object
 7   Credit_amount     1000 non-null   int64
 8   Duration          1000 non-null   int64
 9   Purpose           1000 non-null   object
 10  Risk              1000 non-null   object
dtypes: int64(5), object(6)
memory usage: 86.1+ KB
```

In [244…
```
sns.countplot("Risk",hue='Checking_account',data=df_credit,color="r")
```

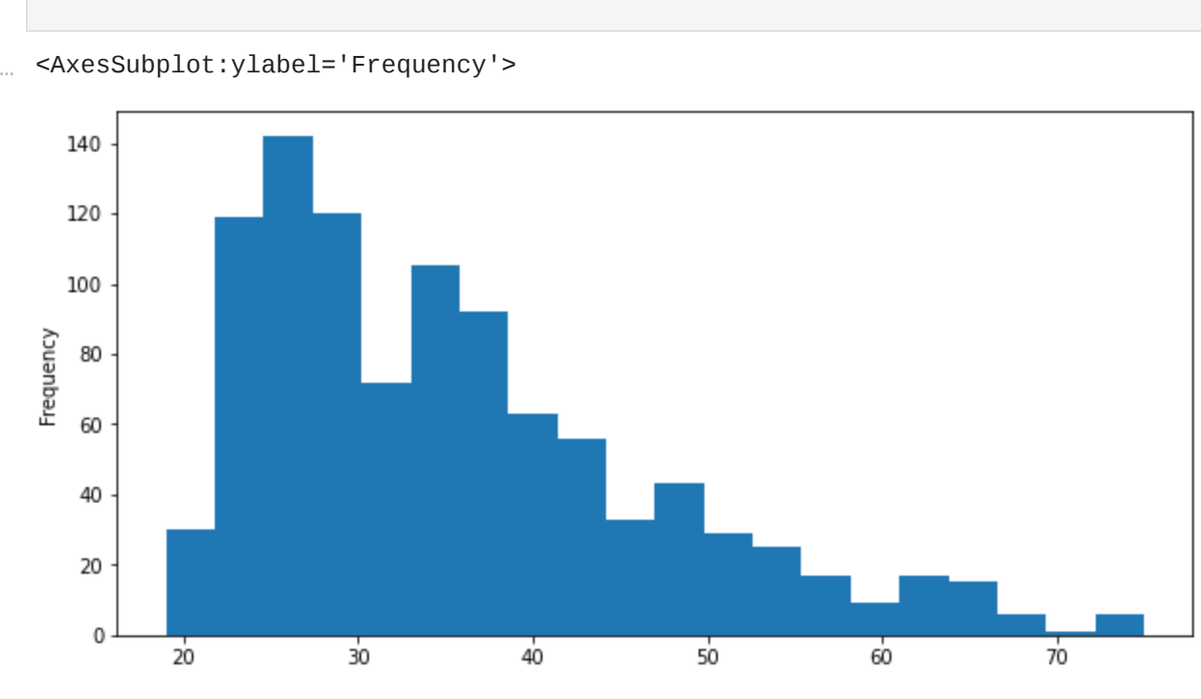
```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[244… <AxesSubplot:xlabel='Risk', ylabel='count'>



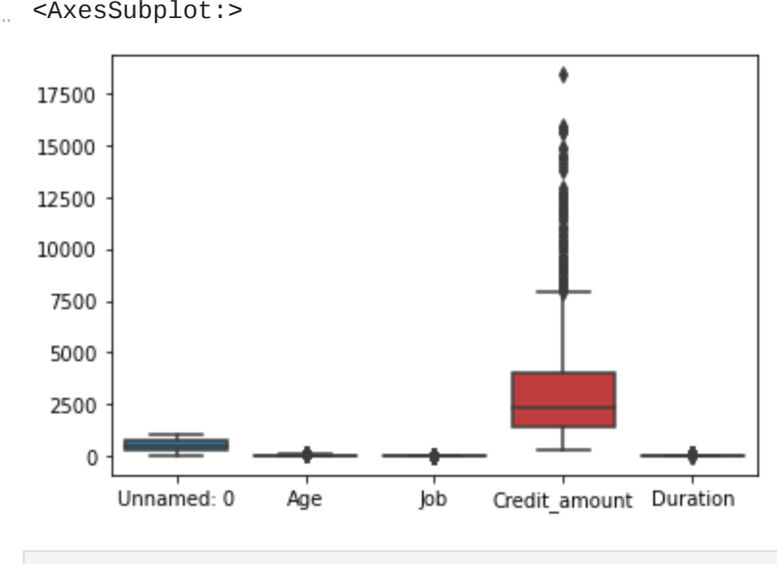
In [245…
```
df_credit['Age'].plot.hist(bins=20,figsize=(10,5))
```

Out[245… <AxesSubplot:ylabel='Frequency'>



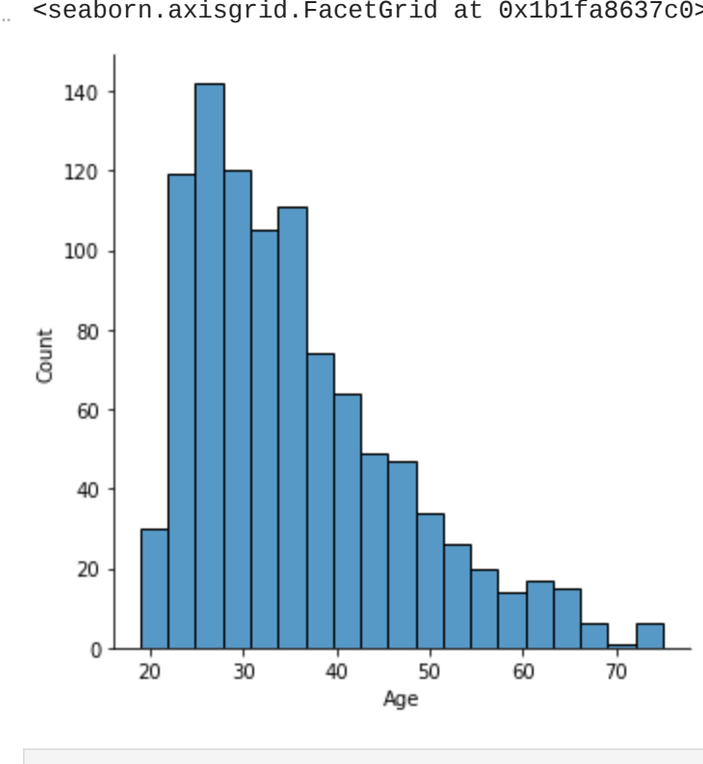
In [246…
```
sns.boxplot(data=df_credit)
```

Out[246… <AxesSubplot:>



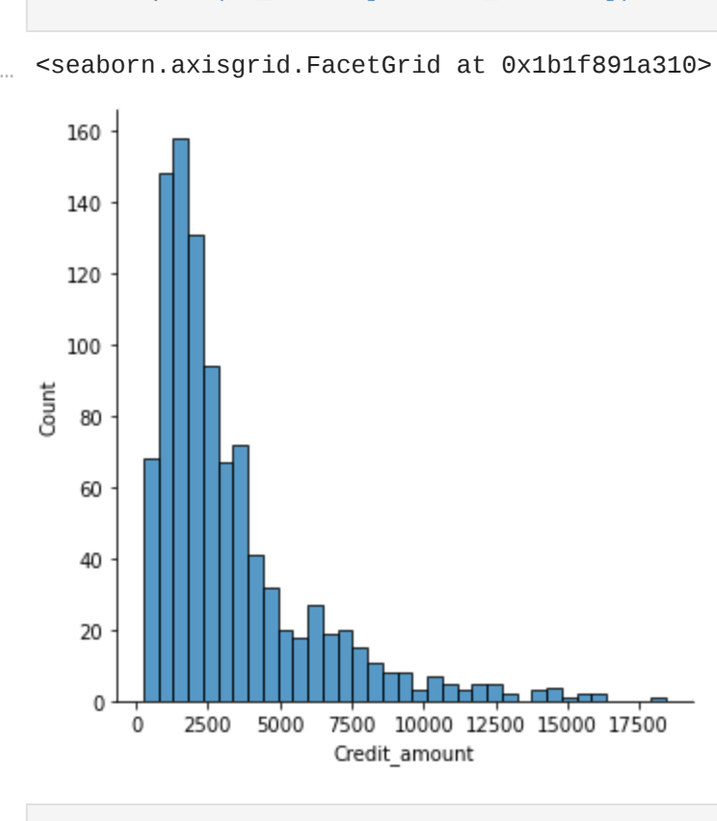
In [247…
```
sns.displot(df_credit['Age'])
```

Out[247… <seaborn.axisgrid.FacetGrid at 0x1b1fa8637c0>



In [248…
```
sns.displot(df_credit['Credit_amount'])
```

Out[248… <seaborn.axisgrid.FacetGrid at 0x1b1f891a310>



In [249…
```
df_credit['Credit_amount'].describe()
```

Out[249…
```
count     1000.000000
mean      3271.258000
std       2822.736876
min        250.000000
25%       1365.500000
50%       2319.500000
75%       3972.250000
max      18424.000000
Name: Credit_amount, dtype: float64
```

In [250…
```
df_credit.boxplot(column="Credit_amount")
```

Out[250… <AxesSubplot:>

In [251…
```
IQR=df_credit.Credit_amount.quantile(0.75)-df_credit.Credit_amount.quantile(0.25)
```

In [252…
```
lower_bridge=df_credit['Credit_amount'].quantile(0.25)-(IQR*1.5)
upper_bridge=df_credit['Credit_amount'].quantile(0.75)+(IQR*1.5)
print(lower_bridge), print(upper_bridge)
```

```
-2544.625
7882.375
```

Out[252… (None, None)

In [253…
```
lower_bridge=df_credit['Credit_amount'].quantile(0.25)-(IQR*3)
upper_bridge=df_credit['Credit_amount'].quantile(0.75)+(IQR*3)
print(lower_bridge), print(upper_bridge)
```

```
-6454.75
11792.5
```

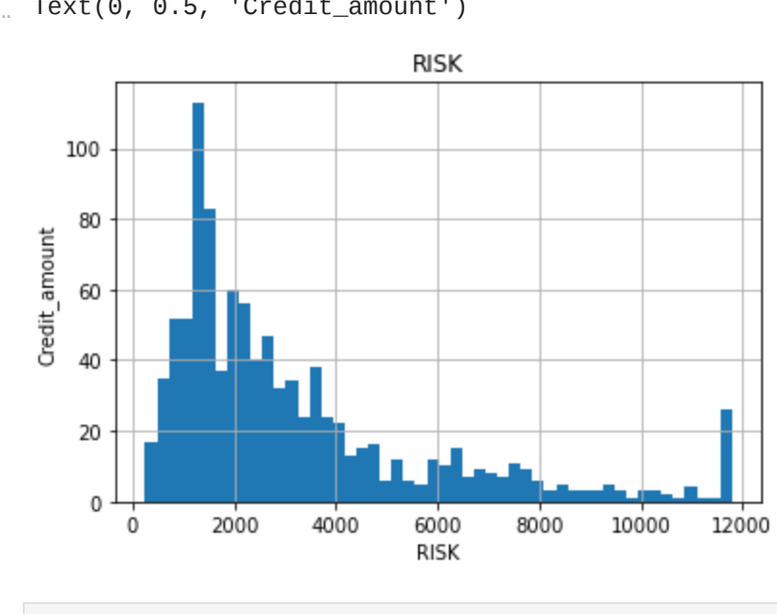
Out[253… (None, None)

In [254…
```
data=df_credit.copy()
```

In [255…
```
data.loc[data['Credit_amount']>=11792.5,'Credit_amount']=11792.5
```

In [256…
```
figure=data.Credit_amount.hist(bins=50)
figure.set_title('RISK')
figure.set_xlabel('RISK')
figure.set_ylabel('Credit_amount')
```

Out[256… Text(0, 0.5, 'Credit_amount')



In [263…
```
Risk_dum=pd.get_dummies(df_credit['Risk'], drop_first=True) [:5]
```

In [264…
```
df_credit.head()
```

Out[264…

	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking_account	Credit_amount	Duration	Purpose	Risk
0	0	67	male	2	own	little	little	1169	6	radio/TV	good
1	1	22	female	2	own	little	moderate	5951	48	radio/TV	bad
2	2	49	male	1	own	little	little	2096	12	education	good
3	3	45	male	2	free	little	little	7882	42	furniture/equipment	good
4	4	53	male	2	free	little	little	4870	24	car	bad

In [275…
```
df_credit['Risk']=pd.get_dummies(df_credit['Risk'],dtype=int, drop_first=True)
```

In [278…
```
df_credit.head()
```

Out[278…

	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking_account	Credit_amount	Duration	Purpose	Risk
0	0	67	1	2	own	little	little	1169	6	radio/TV	1
1	1	22	0	2	own	little	moderate	5951	48	radio/TV	0
2	2	49	1	1	own	little	little	2096	12	education	1
3	3	45	1	2	free	little	little	7882	42	furniture/equipment	1
4	4	53	1	2	free	little	little	4870	24	car	0

In [277…
```
df_credit['Sex']=pd.get_dummies(df_credit['Sex'],dtype=int, drop_first=True)
```

In [279…
```
df_credit.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        1000 non-null   int64
 1   Age               1000 non-null   int64
 2   Sex               1000 non-null   int32
 3   Job               1000 non-null   int64
 4   Housing           1000 non-null   object
 5   Saving accounts   1000 non-null   object
 6   Checking_account  1000 non-null   object
 7   Credit_amount     1000 non-null   int64
 8   Duration          1000 non-null   int64
 9   Purpose           1000 non-null   object
 10  Risk              1000 non-null   int32
dtypes: int32(2), int64(5), object(4)
memory usage: 78.2+ KB
```

In [281…
```
from sklearn import preprocessing
```

In [283…
```
label_encoder=preprocessing.LabelEncoder()
```

In [292…
```
df_credit['Housing']=label_encoder.fit_transform(df_credit['Housing'])
df_credit['Saving accounts']=label_encoder.fit_transform(df_credit['Saving accounts'])
df_credit['Checking_account']=label_encoder.fit_transform(df_credit['Checking_account'])
df_credit['Purpose']=label_encoder.fit_transform(df_credit['Purpose'])
```

In [293…
```
df_credit.head()
```

Out[293…

	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking_account	Credit_amount	Duration	Purpose	Risk
0	0	67	1	2	1	0	0	1169	6	5	1
1	1	22	0	2	1	0	1	5951	48	5	0
2	2	49	1	1	1	0	0	2096	12	3	1
3	3	45	1	2	0	0	0	7882	42	4	1
4	4	53	1	2	0	0	0	4870	24	1	0

In [294…
```
x=df_credit.drop('Risk',axis=1,)
y=df_credit['Risk']
```

In [295…
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=0)
```

In [296…
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression()
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
y_pred1=classifier.predict_proba(X_test)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [297…
```
from sklearn.metrics import accuracy_score,roc_auc_score
print("Accuracy_score: {}".format(accuracy_score(y_test,y_pred)))
print("roc_auc_score: {}".format(roc_auc_score(y_test,y_pred1[:,1])))
```

```
Accuracy_score: 0.9939393939393939
roc_auc_score: 0.993920972644377
```

In [ ]: