```
# For using local files from system, import os


import os
import pandas as pd


#Changing directory to our specified location


os.chdir(data_dir)


dat=pd.read_csv("credit_history.csv")
dat.head()
```

	default	amount	grade	years	ownership	income	age
0	0	1000	B	2.0	RENT	19200.0	24
1	1	6500	A	2.0	MORTGAGE	66000.0	28
2	0	2400	A	2.0	RENT	60000.0	36
3	0	10000	C	3.0	RENT	62000.0	24
4	1	4000	C	2.0	RENT	20000.0	28

```
#Checking for null values


dat.isnull().sum()
```

```
default        0
amount         0
grade          0
years        279
ownership      0
income         0
age            0
dtype: int64
```

```
# Years column have null values which can be imputed by median here.


dat['years'].describe()
```

```
count    7448.000000
mean        6.086332
std         6.700758
min         0.000000
25%         2.000000
50%         4.000000
75%         8.000000
max        62.000000
Name: years, dtype: float64
```

```
dat['years'].fillna(4,inplace=True)
```

```
C:\Users\abhis\AppData\Local\Temp\ipykernel_6336\3242087125.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  dat['years'].fillna(4,inplace=True)
```

```
dat.isnull().sum()
```

```
default      0
amount       0
grade        0
years        0
ownership    0
income       0
age          0
dtype: int64
```

```
dat.head()
```

	default	amount	grade	years	ownership	income	age
0	0	1000	B	2.0	RENT	19200.0	24
1	1	6500	A	2.0	MORTGAGE	66000.0	28
2	0	2400	A	2.0	RENT	60000.0	36
3	0	10000	C	3.0	RENT	62000.0	24
4	1	4000	C	2.0	RENT	20000.0	28

```
#no null values, now we remove default column for making it predictor matrix

X=dat.drop("default",axis=1)

# one hot encoding categorical variables

X=pd.get_dummies(X)

X.head()
```

	amount	years	income	age	grade_A	grade_B	grade_C	grade_D	grade_E	grade_F	grade_G	ownership_MORTGAGE	ownership_OTHER	owners
0	1000	2.0	19200.0	24	False	True	False	False	False	False	False	False	False	
1	6500	2.0	66000.0	28	True	False	False	False	False	False	False	True	False	
2	2400	2.0	60000.0	36	True	False	False	False	False	False	False	False	False	
3	10000	3.0	62000.0	24	False	False	True	False	False	False	False	False	False	
4	4000	2.0	20000.0	28	False	False	True	False	False	False	False	False	False	

```
#Assigning default column as target variable

y=dat['default']

#Splitting predictor matrix into 80-20 ratio into training and testing set.

import sklearn.model_selection as model_selection
X_train,X_test,y_train,y_test=model_selection.train_test_split(X,y,test_size=0.2,random_state=200)

import sklearn.tree as tree
clf=tree.DecisionTreeClassifier(max_depth=3,random_state=200)
clf.fit(X_train,y_train)
clf.score(X_test,y_test)
```

```
0.6274256144890039
```

```
# We get accuracy score as 0.63(approx) with above parameters, now we try to get best parameters
```

Grid Search-Cross Validation

```
clf=tree.DecisionTreeClassifier(max_depth=3,random_state=200)

mod=model_selection.GridSearchCV(clf,param_grid={'max_depth':[2,3,4,5,6]})
mod.fit(X_train,y_train)
```

```
GridSearchCV
best_estimator_: DecisionTreeClassifier
    DecisionTreeClassifier
```

```
mod.best_estimator_
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2, random_state=200)
```

```
mod.best_score_
```

```
0.6314528049645112
```

We can see, for depth=2, we get the best accuracy (0.63). THis is how a simple classification model can be built from decision trees.

Start coding or generate with AI.