Predicting whether a customer would be signing a loan or not

```
from google.colab import files
uploaded = files.upload()
```

Importing modules

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading the dataset

```
import io
dataset = pd.read_csv("/content/Project-9.csv")
dataset
```

	entry_id	age	pay_schedule	home_owner	income	months_employed	years_employed	current_address_year	personal
0	7629673	40	bi-weekly	1	3135	0	3	3	
1	3560428	61	weekly	0	3180	0	6	3	
2	6934997	23	weekly	0	1540	6	0	0	
3	5682812	40	bi-weekly	0	5230	0	6	1	
4	5335819	33	semi-monthly	0	3590	0	5	2	
...	...	...	...	...	...	...	...	...	
17903	9949728	31	monthly	0	3245	0	5	3	
17904	9442442	46	bi-weekly	0	6525	0	2	1	
17905	9857590	46	weekly	0	2685	0	5	1	
17906	8708471	42	bi-weekly	0	2515	0	3	5	
17907	1498559	29	weekly	1	2665	0	4	10	

17908 rows × 21 columns

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17908 entries, 0 to 17907
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   entry_id              17908 non-null  int64  
 1   age                   17908 non-null  int64  
 2   pay_schedule          17908 non-null  object 
 3   home_owner            17908 non-null  int64  
 4   income                17908 non-null  int64  
 5   months_employed       17908 non-null  int64  
 6   years_employed        17908 non-null  int64  
 7   current_address_year  17908 non-null  int64  
 8   personal_account_m    17908 non-null  int64  
 9   personal_account_y    17908 non-null  int64  
 10  has_debt              17908 non-null  int64  
 11  amount_requested      17908 non-null  int64  
 12  risk_score            17908 non-null  int64  
 13  risk_score_2          17908 non-null  float64
 14  risk_score_3          17908 non-null  float64
 15  risk_score_4          17908 non-null  float64
 16  risk_score_5          17908 non-null  float64
 17  ext_quality_score     17908 non-null  float64
 18  ext_quality_score_2   17908 non-null  float64
 19  inquiries_last_month  17908 non-null  int64  
 20  e_signed              17908 non-null  int64  
dtypes: float64(6), int64(14), object(1)
memory usage: 2.9+ MB
```

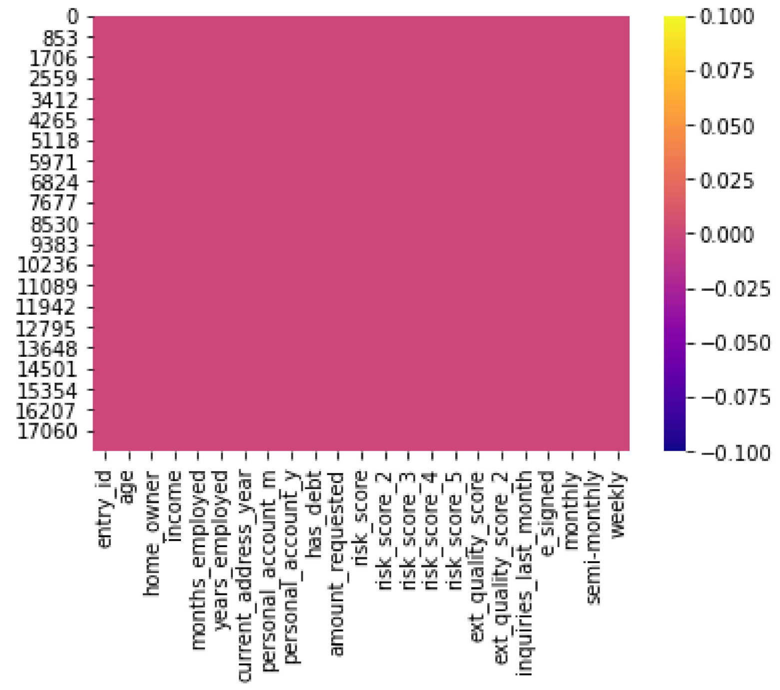
```
dataset.shape
```

```
(17908, 21)
```

Ensuring there's no missing data

```
sns.heatmap(dataset.isnull(), cmap='plasma')
```

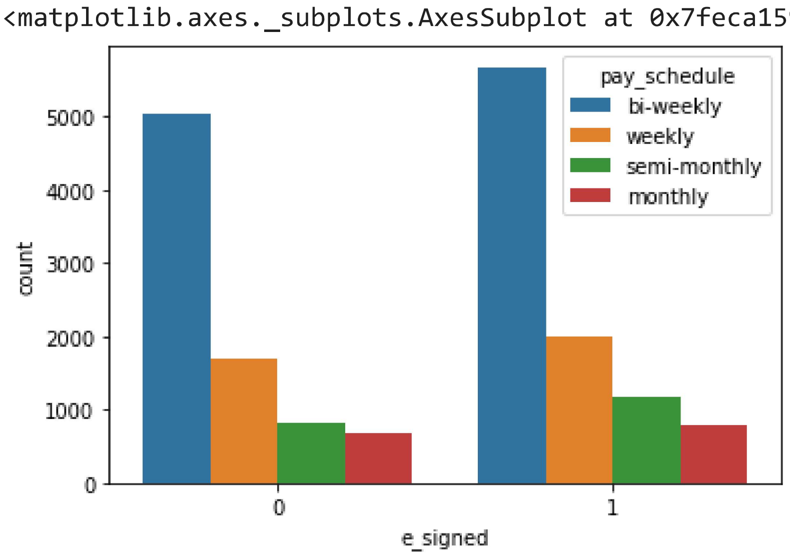
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fec9bd031d0>
```



Data visualization

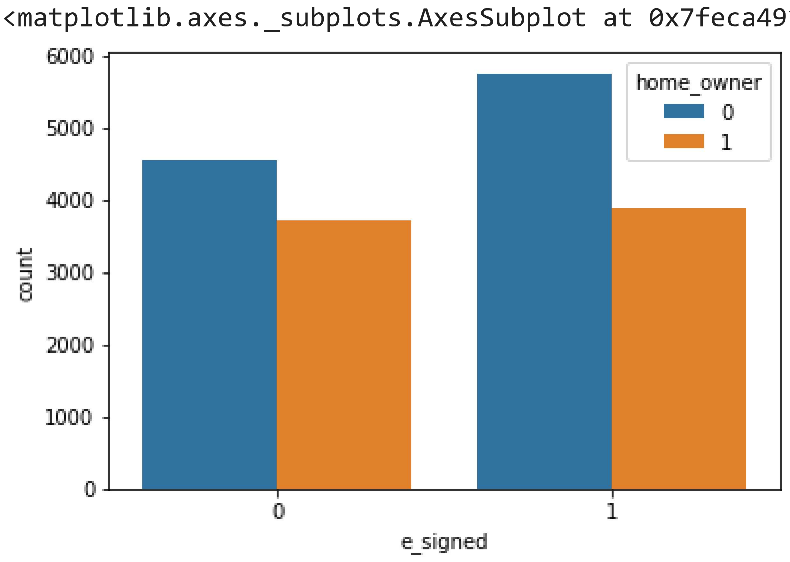
```
sns.countplot(x='e_signed', hue='pay_schedule', data= dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feca1598358>
```



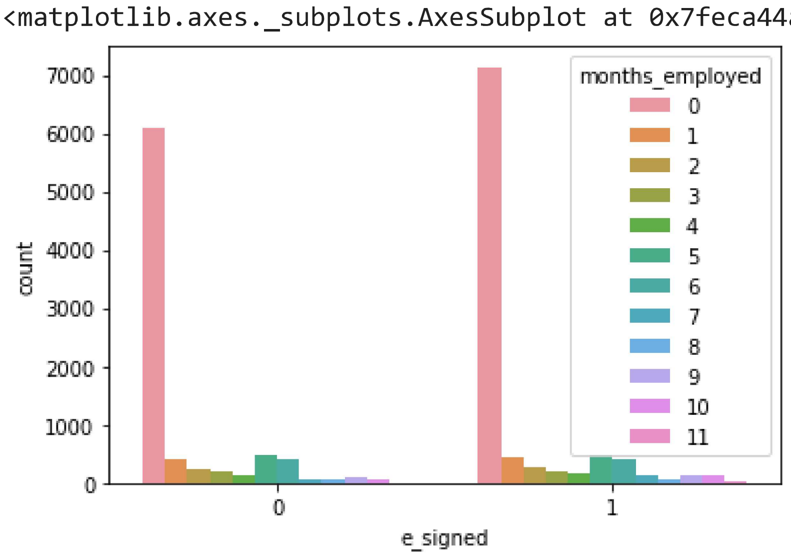

```
sns.countplot(x='e_signed', hue='home_owner', data= dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feca4916668>
```




```
sns.countplot(x='e_signed', hue='months_employed', data= dataset)
```

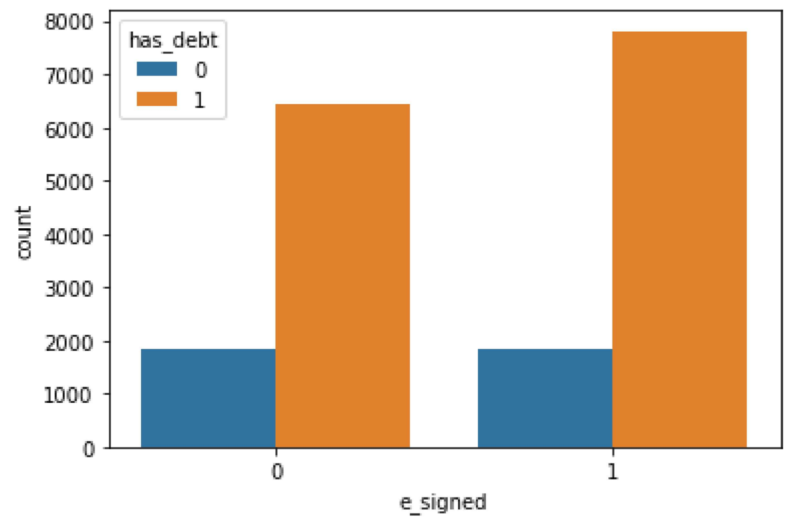
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feca44aab38>
```




```
sns.countplot(x='e_signed', hue='has_debt', data= dataset)
```

```
sns.countplot(x='e_signed', hue='has_debt', data=dataset)
```

<matplotlib.axes._subplots.AxesSubplot at 0x2c93b598>



Encoding categorical data

```
pay = pd.get_dummies(dataset['pay_schedule'], drop_first = True)
pay
```

	monthly	semi-monthly	weekly
0	0	0	0
1	0	0	1
2	0	0	1
3	0	0	0
4	0	1	0
...	...	...	...
17903	1	0	0
17904	0	0	0
17905	0	0	1
17906	0	0	0
17907	0	0	1

17908 rows × 3 columns

```
dataset = pd.concat([dataset,pay], axis=1)
dataset.drop(['pay_schedule'], axis =1 ,inplace = True)
dataset
```

	entry_id	age	home_owner	income	months_employed	years_employed	current_address_year	personal_account_m	pe
0	7629673	40	1	3135	0	3	3	6	
1	3560428	61	0	3180	0	6	3	2	
2	6934997	23	0	1540	6	0	0	7	
3	5682812	40	0	5230	0	6	1	2	
4	5335819	33	0	3590	0	5	2	2	
...	...	...	...	...	...	...	...	...	
17903	9949728	31	0	3245	0	5	3	2	
17904	9442442	46	0	6525	0	2	1	3	
17905	9857590	46	0	2685	0	5	1	1	
17906	8708471	42	0	2515	0	3	5	6	
17907	1498559	29	1	2665	0	4	10	4	

17908 rows × 23 columns

Extracting features and labels

```
x = dataset.drop('e_signed', axis= 1)
x
```

	entry_id	age	home_owner	income	months_employed	years_employed	current_address_year	personal_account_m	pe
0	7629673	40	1	3135	0	3	3	6	
1	3560428	61	0	3180	0	6	3	2	
2	6934997	23	0	1540	6	0	0	7	
3	5682812	40	0	5230	0	6	1	2	
4	5335819	33	0	3590	0	5	2	2	
...	...	...	...	...	...	...	...	...	
17903	9949728	31	0	3245	0	5	3	2	
17904	9442442	46	0	6525	0	2	1	3	
17905	9857590	46	0	2685	0	5	1	1	
17906	8708471	42	0	2515	0	3	5	6	
17907	1498559	29	1	2665	0	4	10	4	

```
y = dataset['e_signed']
y
```

```
0        1
1        0
2        0
3        1
4        0
        ..
17903    0
17904    0
17905    0
17906    1
17907    1
Name: e_signed, Length: 17908, dtype: int64
```

Splitting the dataset into Training set and Test set

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/3, random_state = 7)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)
```

```
y_pred
```

```
array([0, 1, 1, ..., 1, 1, 1])
```

```
from sklearn.metrics import accuracy_score
acc_score = accuracy_score(y_test, y_pred)
print(acc_score)
```

```
0.5785594639865996
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion = 'entropy', random_state = 7)
dtc.fit(x_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None
```

```
y_pred = dtc.predict(x_test)
y_pred
```

```
array([0, 1, 1, ..., 1, 0, 0], dtype=int64)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
acc_score = accuracy_score(y_test, y_pred)
print(acc_score)
```

```
[[1172 1554]
 [ 962 2282]]
              precision    recall  f1-score   support

           0       0.55      0.43      0.48      2726
           1       0.59      0.70      0.64      3244

    accuracy                           0.58      5970
   macro avg       0.57      0.57      0.56      5970
weighted avg       0.57      0.58      0.57      5970

0.5785594639865996
```

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train, y_train)
y_pred = nb.predict(x_test)
```

```
y_pred
```

```
array([0, 1, 1, ..., 1, 1, 1])
```

```
from sklearn import metrics
print("Accuracy: ", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy:  0.5753768844221105
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 30, random_state = 7)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

```
[[1598 1128]
 [1134 2110]]
              precision    recall  f1-score   support

           0       0.58      0.59      0.59      2726
           1       0.65      0.65      0.65      3244

    accuracy                           0.62      5970
   macro avg       0.62      0.62      0.62      5970
weighted avg       0.62      0.62      0.62      5970

0.6211055276381909
```