CS381/780 Data Analytics Final Project

Due on 12/13/2021 23:59 pm

In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
```

In [4]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

In [5]:

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

Dataset is based on the follwoing

[https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data) (https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data))](https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data))

Attribute Information:

Attribute 1: (qualitative) Status of existing checking account A11 : ... < 0 DM A12 : 0 <= ... < 200 DM A13 : ... >= 200 DM / salary assignments for at least 1 year A14 : no checking account

Attribute 2: (numerical) Duration in month

Attribute 3: (qualitative) Credit history A30 : no credits taken/ all credits paid back duly A31 : all credits at this bank paid back duly A32 : existing credits paid back duly till now A33 : delay in paying off in the past A34 : critical account/ other credits existing (not at this bank)

Attribute 4: (qualitative) Purpose A40 : car (new) A41 : car (used) A42 : furniture/equipment A43 : radio/television A44 : domestic appliances A45 : repairs A46 : education A47 : (vacation - does not exist?) A48 : retraining A49 : business A410 : others

Attribute 5: (numerical) Credit amount

Attibute 6: (qualitative) Savings account/bonds A61 : ... < 100 DM A62 : 100 <= ... < 500 DM A63 : 500 <= ... < 1000 DM A64 : .. >= 1000 DM A65 : unknown/ no savings account

Attribute 7: (qualitative) Present employment since A71 : unemployed A72 : ... < 1 year A73 : 1 <= ... < 4 years A74 : 4 <= ... < 7 years A75 : .. >= 7 years

Attribute 8: (numerical) Installment rate in percentage of disposable income

Attribute 9: (qualitative) Personal status and sex A91 : male : divorced/separated A92 : female : divorced/separated/married A93 : male : single A94 : male : married/widowed A95 : female : single

Attribute 10: (qualitative) Other debtors / guarantors A101 : none A102 : co-applicant A103 : guarantor

Attribute 11: (numerical) Present residence since

Attribute 12: (qualitative) Property A121 : real estate A122 : if not A121 : building society savings agreement/ life insurance A123 : if not A121/A122 : car or other, not in attribute 6 A124 : unknown / no property

Attribute 13: (numerical) Age in years

Attribute 14: (qualitative) Other installment plans A141 : bank A142 : stores A143 : none

Attribute 15: (qualitative) Housing A151 : rent A152 : own A153 : for free

Attribute 16: (numerical) Number of existing credits at this bank

Attribute 17: (qualitative) Job A171 : unemployed/ unskilled - non-resident A172 : unskilled - resident A173 : skilled employee / official A174 : management/ self-employed/ highly qualified employee/ officer

Attribute 18: (numerical) Number of people being liable to provide maintenance for

Attribute 19: (qualitative) Telephone A191 : none A192 : yes, registered under the customers name

Your task in the final project is build the best predictive model to classify if a loan will carry good or bad credit risks. The focus should be in identifying bad risk loans

- Try at least two of the models (Logistic, SVM, Naive Bayes, Decision Tree and Random Forecast) that we have covered in class.
- Do not use any models that we have not covered in class.
-
- Answer the question whether past credit history will be an important factor or not.

Make sure your work include the following steps

- EDA (chekcing missing values, removing outliers)
- performed basic exploration of relationship, with plots and graphs
- separated data set into training and testing
- setup dummy variables to take care categorical variables
- normalize numerical features if needed
- tried at least two models and checked their model performance
- performed cross-validations

In [6]:

```
df = pd.read_csv("german_credit_modified.csv")
df.head()
```

Out[6]:

	Checking Account	Duration	Credit History	Purpose	Credit Amount	Saving Account	Employment Status	Installment Rate	Perso Stat
0	A11	6	A34	A43	1169	A65	A75	4	A
1	A12	48	A32	A43	5951	A61	A73	2	A
2	A14	12	A34	A46	2096	A61	A74	2	A
3	A11	42	A32	A42	7882	A61	A74	2	A
4	A11	24	A33	A40	4870	A61	A73	3	A

In [7]:

```
df['Risk'] = df['Risk'].apply(lambda x: 'good' if x == 1 else 'bad')
```

In [8]:

```
df.head()
```

Out[8]:

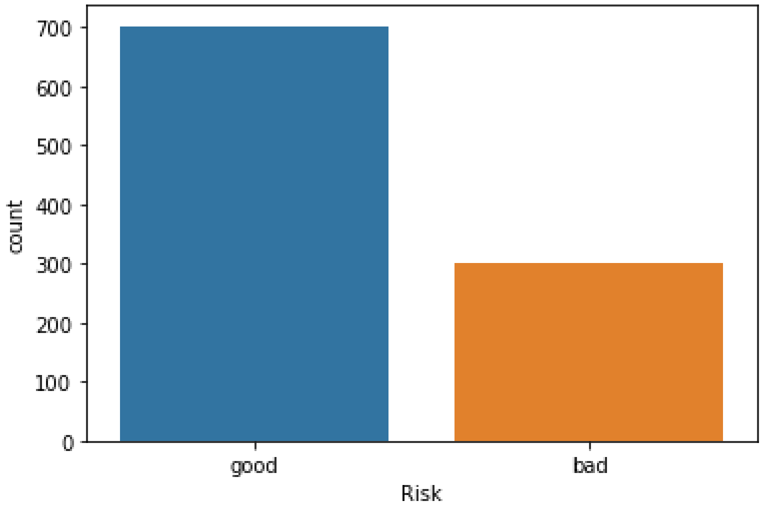
	Checking Account	Duration	Credit History	Purpose	Credit Amount	Saving Account	Employment Status	Installment Rate	Perso Stat
0	A11	6	A34	A43	1169	A65	A75	4	A
1	A12	48	A32	A43	5951	A61	A73	2	A
2	A14	12	A34	A46	2096	A61	A74	2	A
3	A11	42	A32	A42	7882	A61	A74	2	A
4	A11	24	A33	A40	4870	A61	A73	3	A

In [9]:

```
sns.countplot(df['Risk'])
```

Out[9]:

<AxesSubplot:xlabel='Risk', ylabel='count'>



The original dataset is hard to understand. So we are going to decode the fields to an easier to understand format

In [10]:

```
decode_map = {'A11': 'little', 'A12': 'moderate', 'A13': 'rich', 'A14': 'No Account',
              'A30': 'paid back', 'A31': 'paid back', 'A32': 'paid back',
              'A33': 'delay', 'A34': 'default',
              'A40' : 'car',
              'A41' : 'car',
              'A42' : 'furniture/equipment',
              'A43' : 'radio/television',
              'A44' : 'domestic appliances',
              'A45' : 'repairs',
              'A46' : 'education',
              'A47' : 'vacation',
              'A48' : 'retraining',
              'A49' : 'business',
              'A410' : 'others',
              'A61' :  'little',
              'A62' :  'moderate',
              'A63' :  'quite rich',
              'A64' :  'rich',
              'A65' :  'unknown',

              'A71' : 'unemployed',
              'A72' : '< 1 year',
              'A73' : '1 to <4 years',
              'A74': '4 to <7 years',
              'A75' : '>= 7 years',

              'A91' : 'male   : divorced/separated',
              'A92' : 'female : divorced/separated/married',
              'A93' : 'male   : single',
              'A94' : 'male   : married/widowed',
              'A95' : 'female : single',

              'A101' : 'none',
              'A102' : 'co-applicant',
              'A103' : 'guarantor',

              'A121' : 'real estate',
              'A122' : 'life insurance',
              'A123' : 'car',
              'A124' : 'no property',

              'A141' : 'bank',
              'A142' : 'stores',
              'A143' : 'none',
              'A151' : 'rent',
              'A152' : 'own',
              'A153' : 'for free',


              'A171' : 'unemployed/non-resident',
              'A172' : 'unskilled/resident',
              'A173' : 'skilled employee',
              'A174' : 'management/highly qualified employee',

             }
```

In [11]:

```
for col in df.columns:
    df[col] = df[col].apply(lambda x: decode_map[x] if x in decode_map.keys() else x)
```

In [12]:

```
df.head()
```

Out[12]:

	Checking Account	Duration	Credit History	Purpose	Credit Amount	Saving Account	Employment Status	Installmen Rat
0	little	6	default	radio/television	1169	unknown	>= 7 years	
1	moderate	48	paid back	radio/television	5951	little	1 to <4 years	
2	No Account	12	default	education	2096	little	4 to <7 years	
3	little	42	paid back	furniture/equipment	7882	little	4 to <7 years	
4	little	24	delay	car	4870	little	1 to <4 years	

In [13]:

```
df.shape
```

Out[13]:

(1003, 19)

Now you can start from this dataset

Good Luck !!!

Show all your work below

1

In [14]:

```
df.isnull().any()
```

Out[14]:

```
Checking Account            False
Duration                    False
Credit History               True
Purpose                     False
Credit Amount               False
Saving Account              False
Employment Status           False
Installment Rate            False
Personal Status             False
Guarantors                  False
Years in current address    False
Property                    False
Age                         False
Installment plans           False
Housing                     False
Existing Credits            False
Job                          True
Liable                      False
Risk                        False
dtype: bool
```

In [15]:

```
df = df[pd.notnull(df['Credit History'])]
df = df[pd.notnull(df['Job'])]
```

In [16]:

```
df.isnull().any()
```

Out[16]:

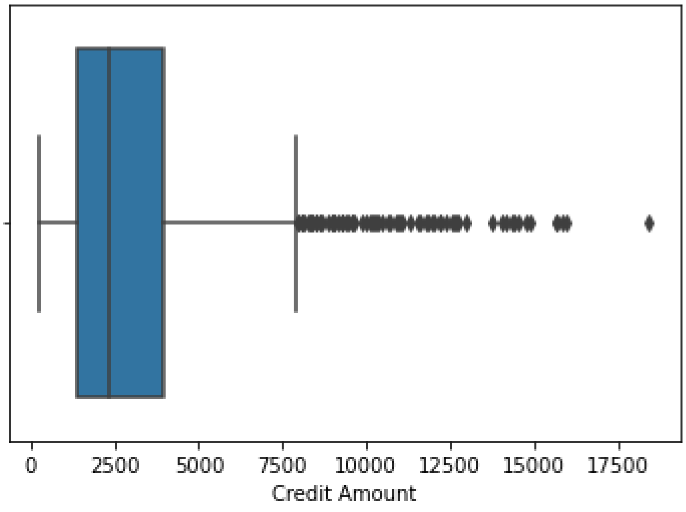
```
Checking Account            False
Duration                    False
Credit History              False
Purpose                     False
Credit Amount               False
Saving Account              False
Employment Status           False
Installment Rate            False
Personal Status             False
Guarantors                  False
Years in current address    False
Property                    False
Age                         False
Installment plans           False
Housing                     False
Existing Credits            False
Job                         False
Liable                      False
Risk                        False
dtype: bool
```

In [17]:

```
sns.boxplot(x=df['Credit Amount'])
```

Out[17]:

<AxesSubplot:xlabel='Credit Amount'>

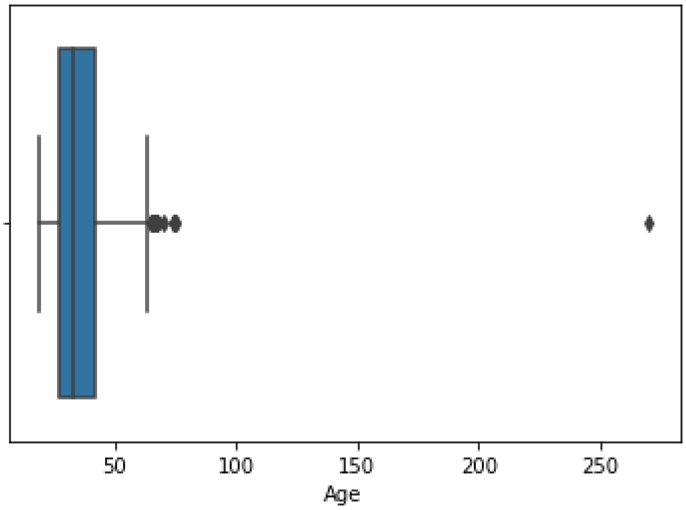


In [18]:

```
sns.boxplot(x=df['Age'])
```

Out[18]:

<AxesSubplot:xlabel='Age'>

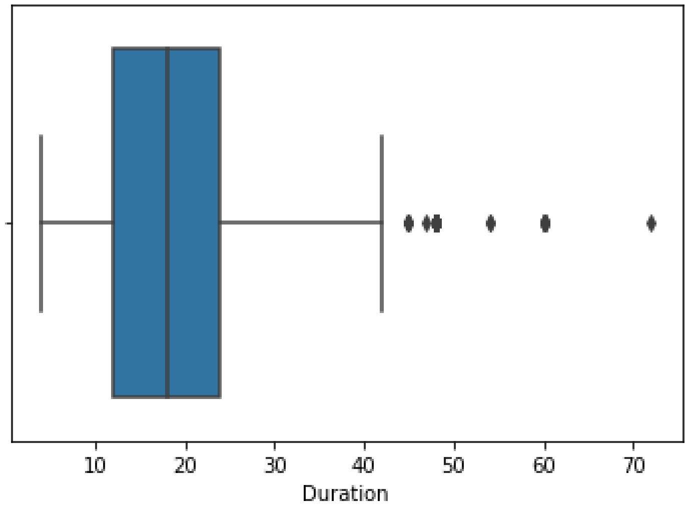


In [19]:

```
sns.boxplot(x=df['Duration'])
```

Out[19]:

<AxesSubplot:xlabel='Duration'>

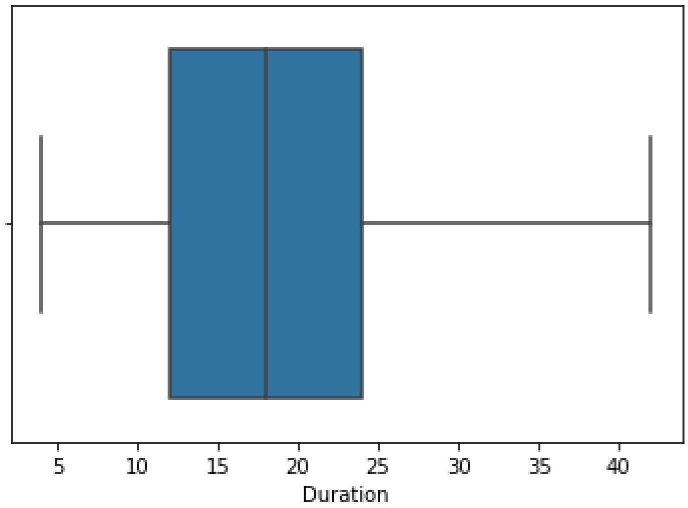


In [20]:

```
df = df[df['Duration'] < 45]
sns.boxplot(x=df['Duration'],data=df)
df.shape
```

Out[20]:

(931, 19)

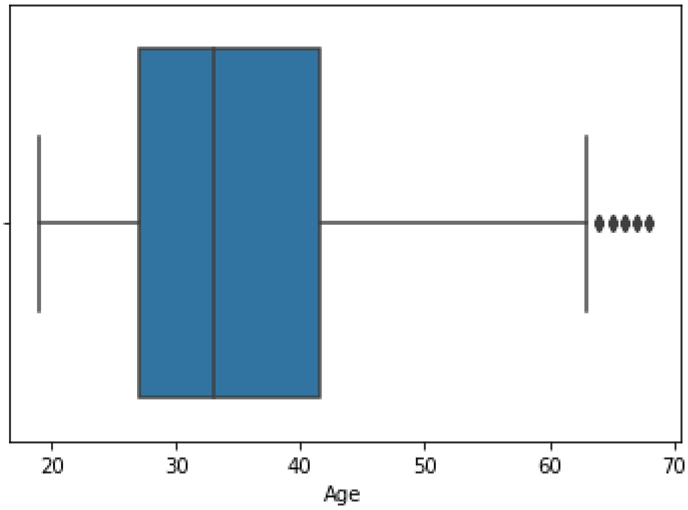


In [21]:

```
df = df[df['Age'] < 70]
sns.boxplot(x=df['Age'],data=df)
df.shape
```

Out[21]:

(923, 19)

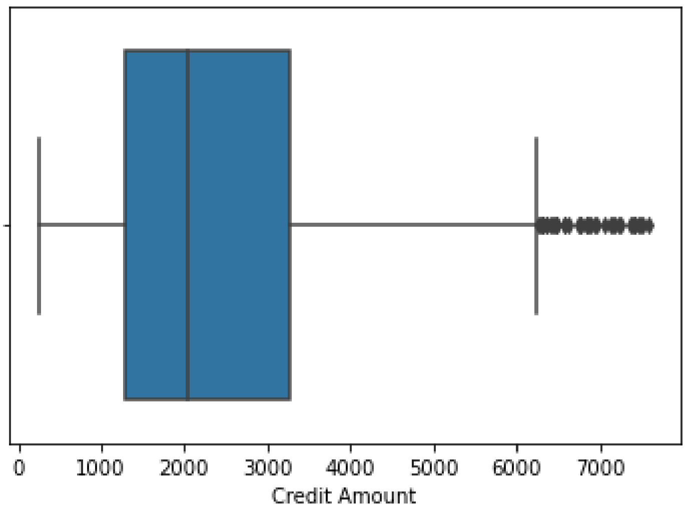


In [22]:

```
df = df[df['Credit Amount'] < 7600]
sns.boxplot(x=df['Credit Amount'],data=df)
df.shape
```

Out[22]:

(869, 19)



In [23]:

```
df.groupby('Risk').mean()
```

Out[23]:

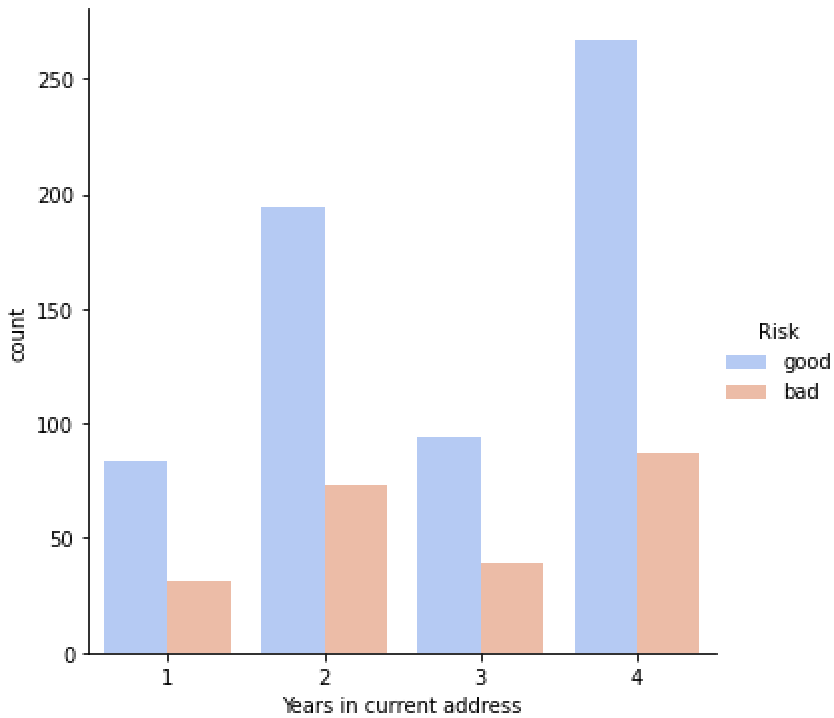
	Duration	Credit Amount	Installment Rate	Years in current address	Age	Existing Credits	Liable
Risk							
bad	20.065217	2444.291304	3.204348	2.791304	33.217391	1.356522	1.16087
good	17.258216	2483.176839	2.965571	2.851330	35.934272	1.427230	1.14554

In [24]:

```
sns.factorplot('Years in current address', kind='count', hue='Risk', data=df,palette='coolwa
```

Out[24]:

<seaborn.axisgrid.FacetGrid at 0x199b4de2940>

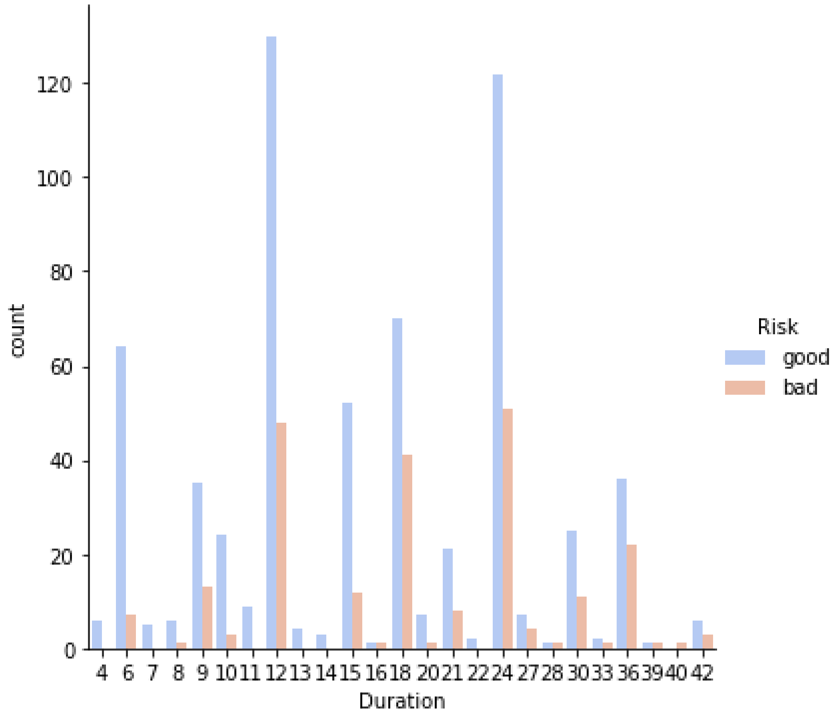


In [25]:

```
sns.factorplot('Duration', kind='count', hue='Risk', data=df,palette='coolwarm')
```

Out[25]:

<seaborn.axisgrid.FacetGrid at 0x199aeb82280>

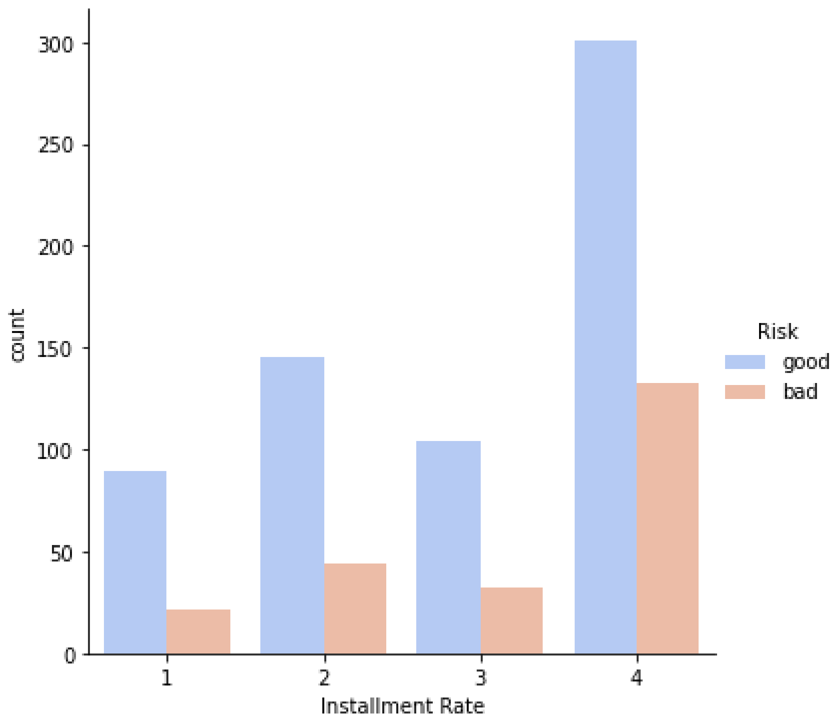


In [26]:

```
sns.factorplot('Installment Rate', kind='count', hue='Risk', data=df,palette='coolwarm')
```

Out[26]:

<seaborn.axisgrid.FacetGrid at 0x199b4c7cee0>

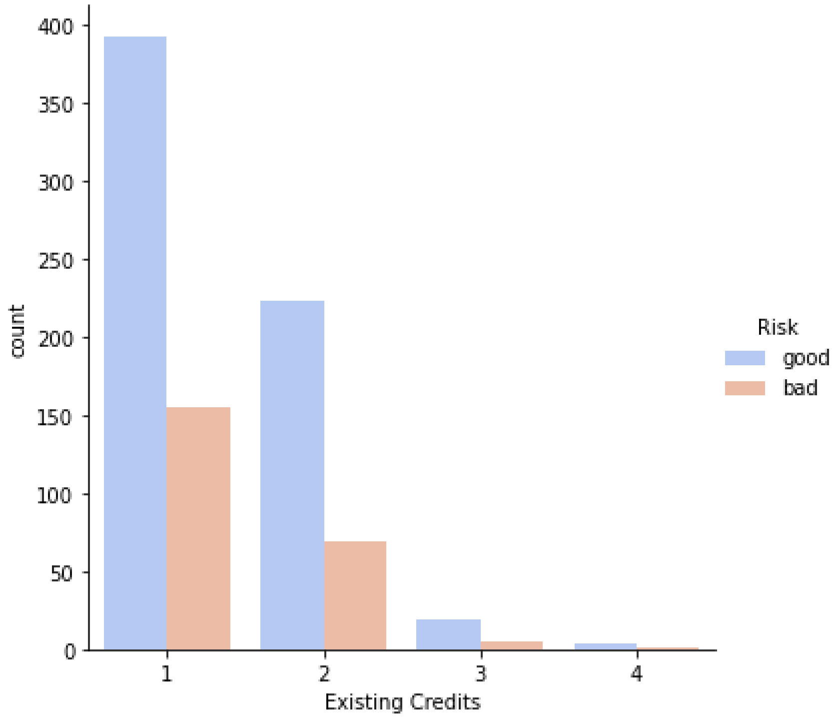


In [27]:

```
sns.factorplot('Existing Credits', kind='count', hue='Risk', data=df,palette='coolwarm')
```

Out[27]:

<seaborn.axisgrid.FacetGrid at 0x199b5009fa0>

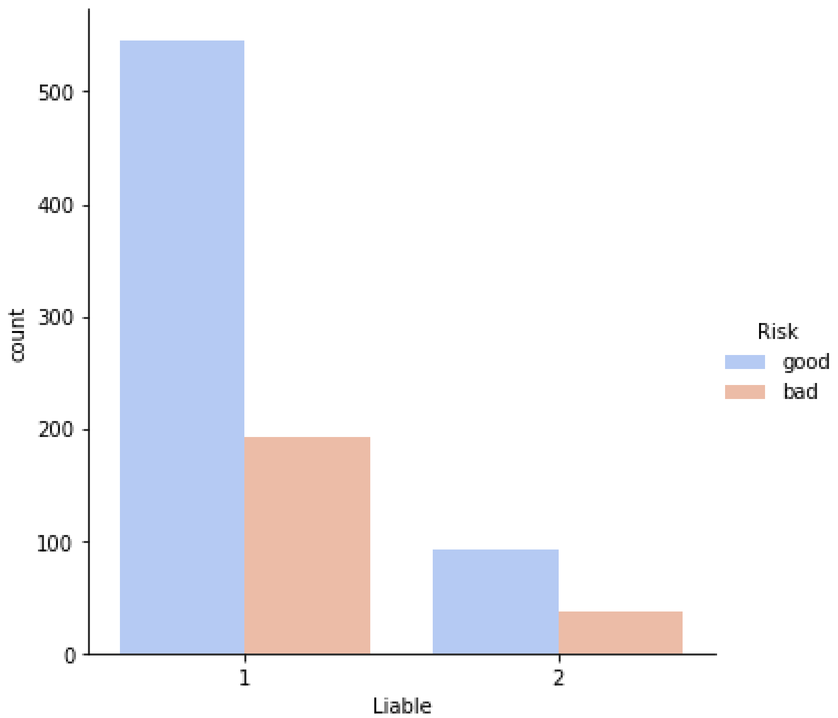


In [28]:

```
sns.factorplot('Liable', kind='count', hue='Risk', data=df,palette='coolwarm')
```

Out[28]:

<seaborn.axisgrid.FacetGrid at 0x199b50010d0>



In [ ]:

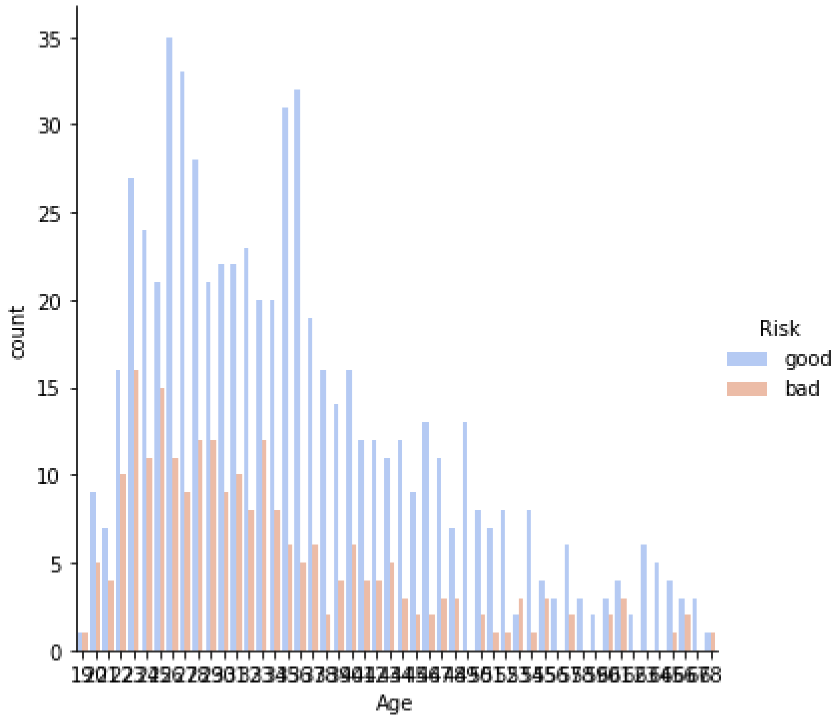
```
1
```

In [29]:

```
sns.factorplot('Age', kind='count', hue='Risk', data=df,palette='coolwarm')
```

Out[29]:

<seaborn.axisgrid.FacetGrid at 0x199b60a49d0>

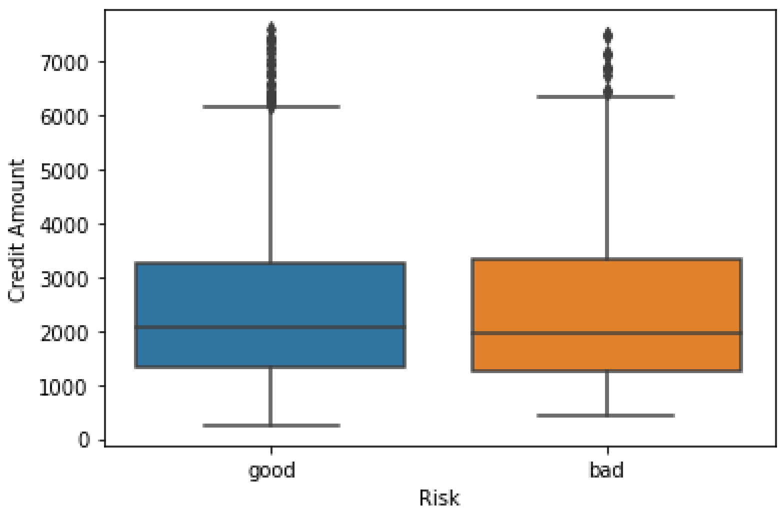


In [30]:

```
sns.boxplot(x='Risk', y = 'Credit Amount', data=df)
```

Out[30]:

<AxesSubplot:xlabel='Risk', ylabel='Credit Amount'>

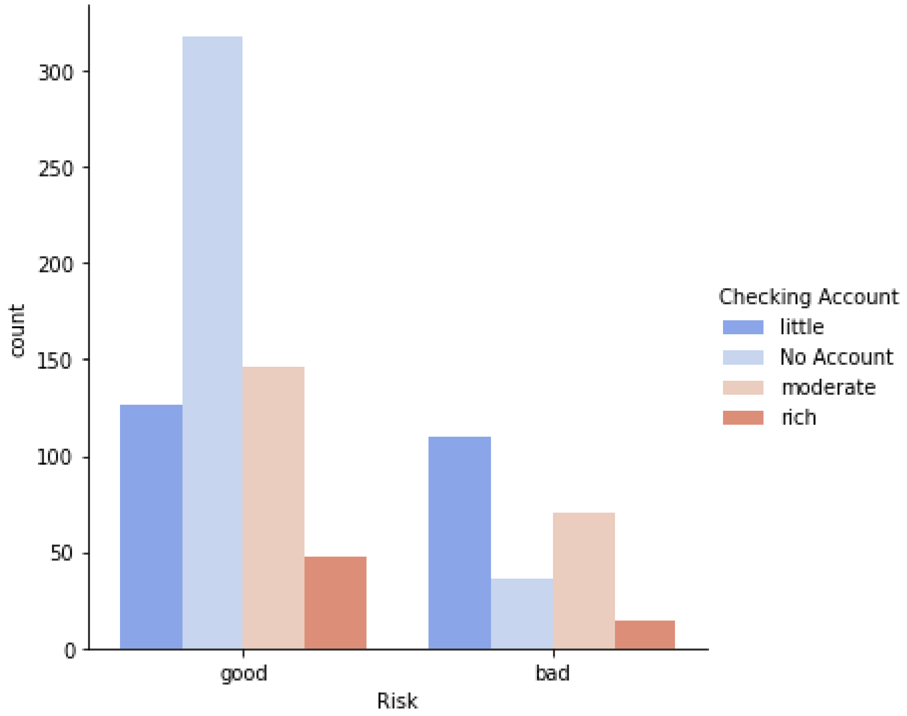


In [31]:

```
sns.factorplot(x='Risk', kind='count', hue='Checking Account', data=df,palette='coolwarm')
```

Out[31]:

<seaborn.axisgrid.FacetGrid at 0x199b4d76c40>

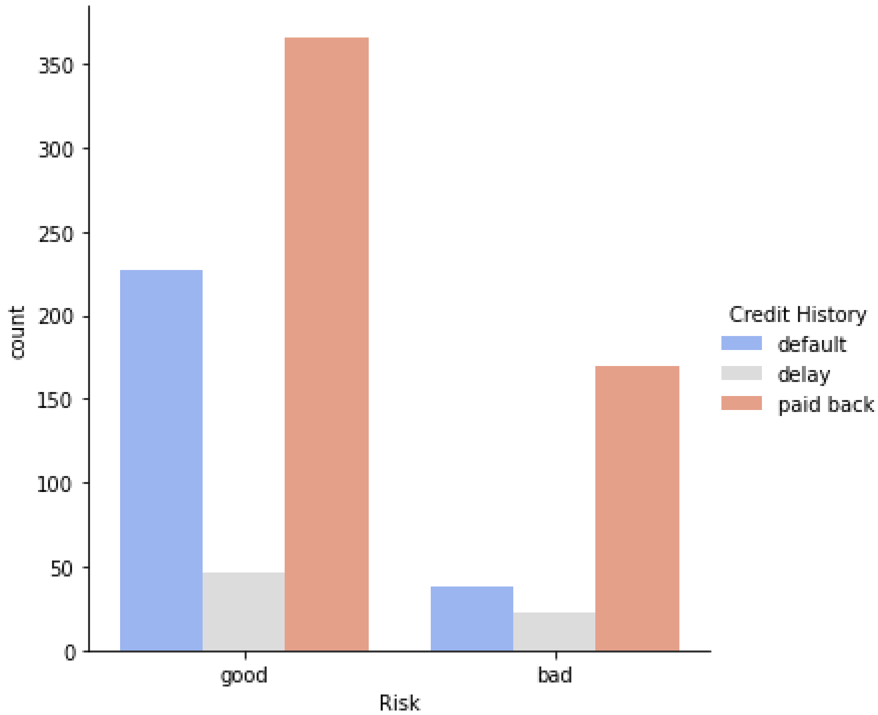


In [32]:

```
sns.factorplot(x='Risk', kind='count', hue='Credit History', data=df,palette='coolwarm')
```

Out[32]:

<seaborn.axisgrid.FacetGrid at 0x199b4c3fc40>

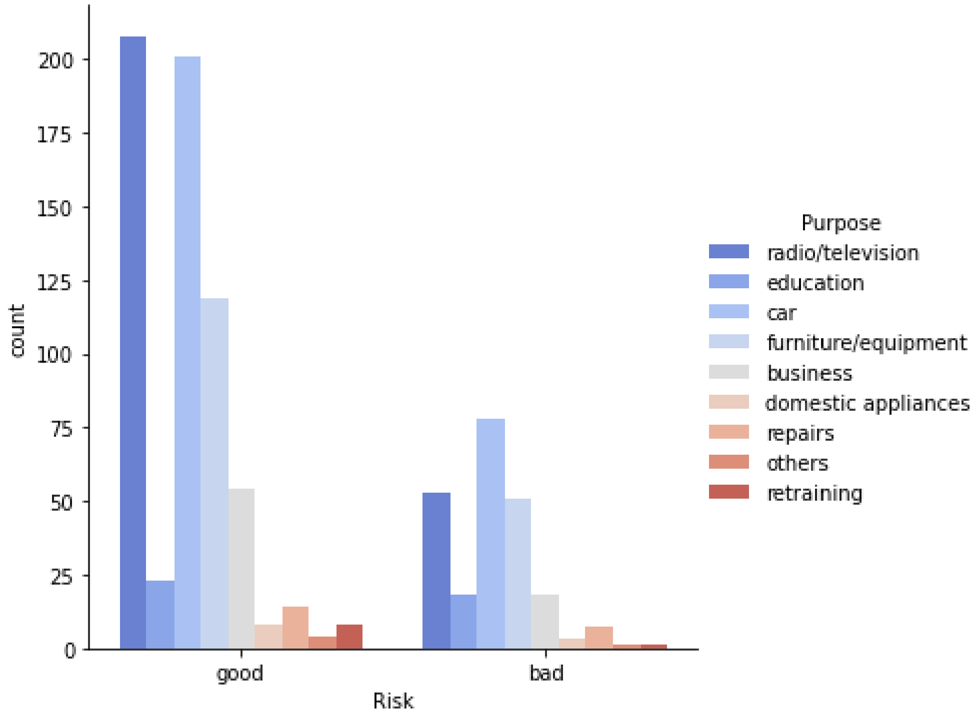


In [33]:

```
sns.factorplot(x='Risk', kind='count', hue='Purpose', data=df,palette='coolwarm')
```

Out[33]:

<seaborn.axisgrid.FacetGrid at 0x199b4d14640>

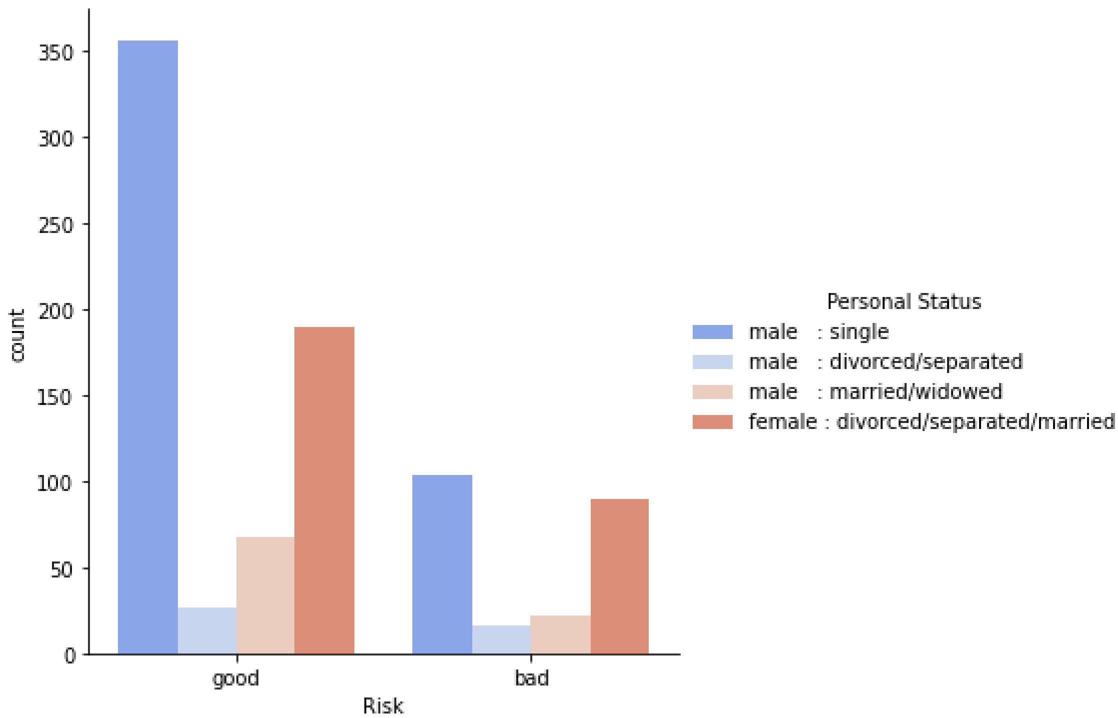


In [34]:

```
sns.factorplot(x='Risk', kind='count', hue='Personal Status', data=df,palette='coolwarm')
```

Out[34]:

<seaborn.axisgrid.FacetGrid at 0x199b6310820>

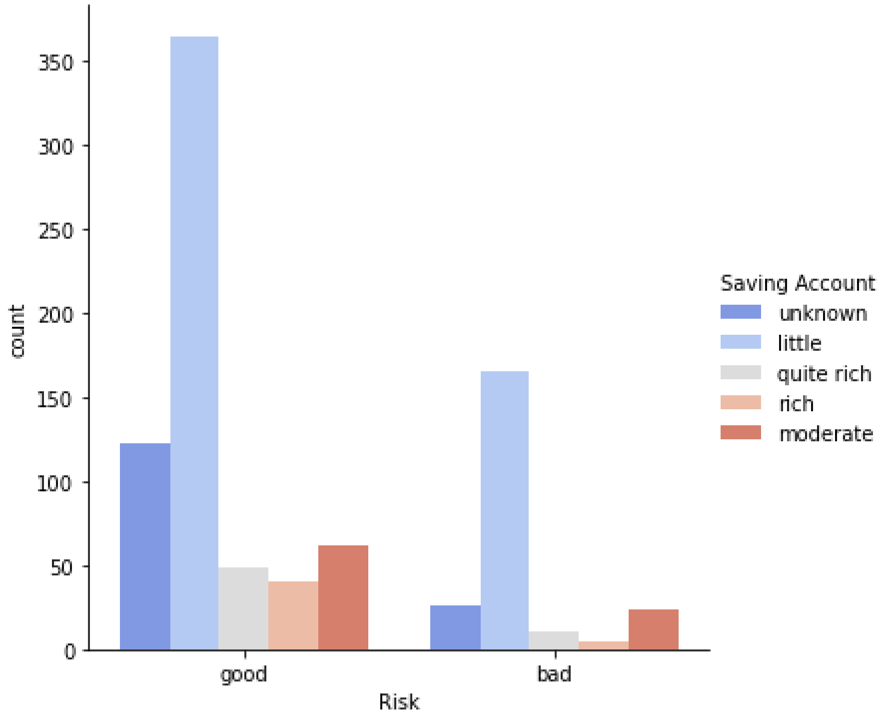


In [35]:

```
sns.factorplot(x='Risk', kind='count', hue='Saving Account', data=df,palette='coolwarm')
```

Out[35]:

<seaborn.axisgrid.FacetGrid at 0x199b637ff10>

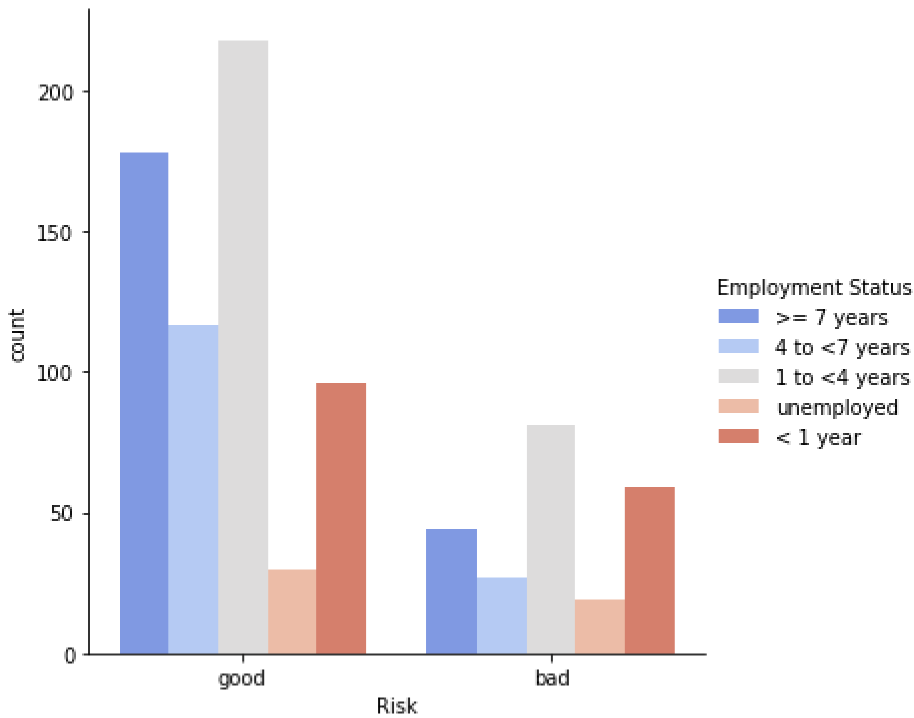


In [36]:

```
sns.factorplot(x='Risk', kind='count', hue='Employment Status', data=df,palette='coolwarm')
```

Out[36]:

<seaborn.axisgrid.FacetGrid at 0x199b4f6e7f0>

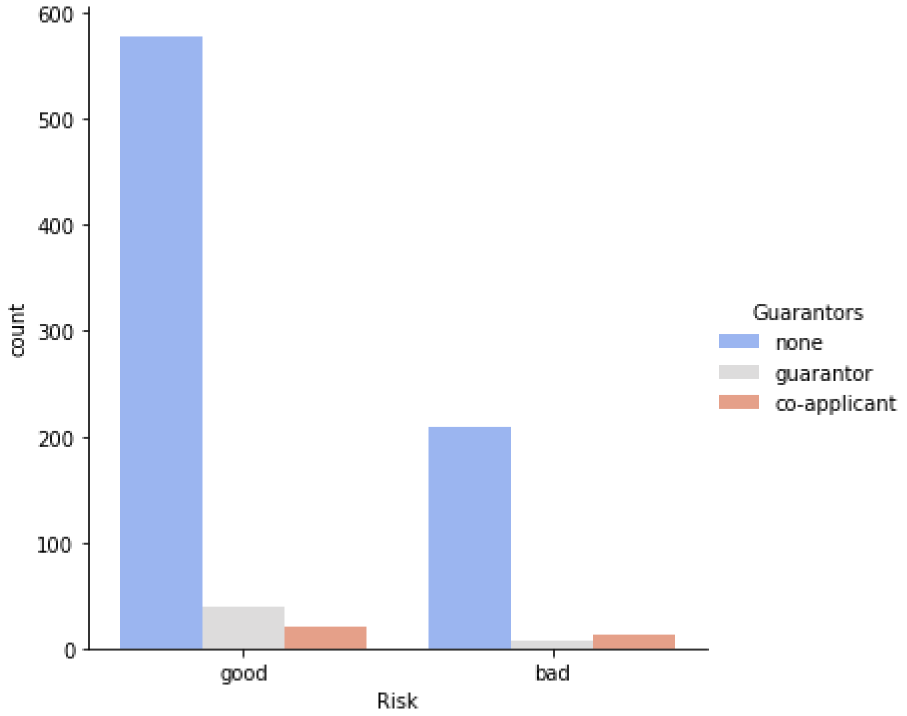


In [37]:

```
sns.factorplot(x='Risk', kind='count', hue='Guarantors', data=df,palette='coolwarm')
```

Out[37]:

<seaborn.axisgrid.FacetGrid at 0x199b4dc97c0>

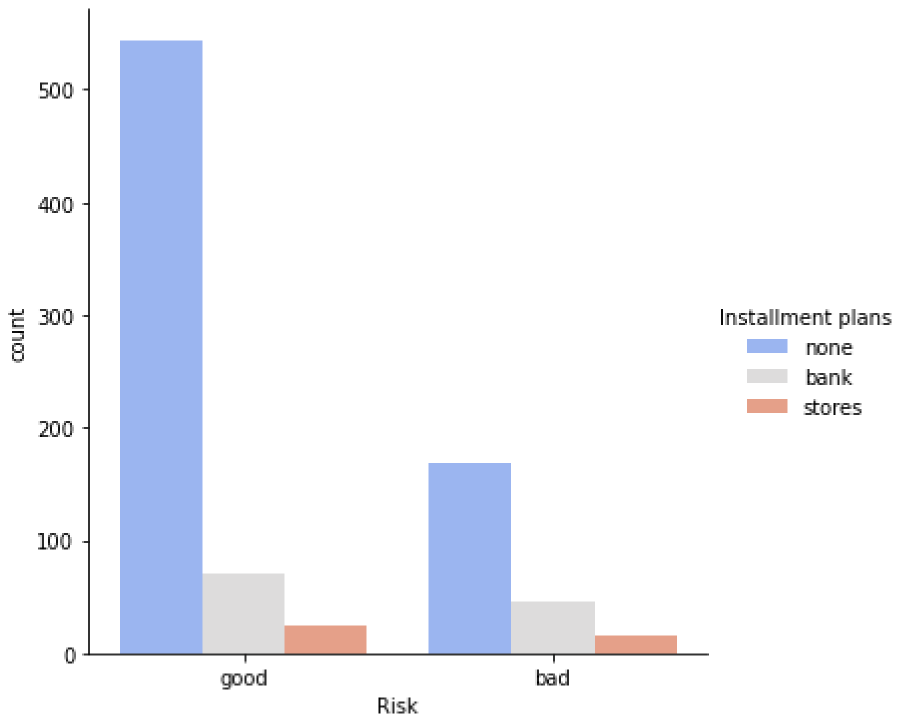


In [38]:

```
sns.factorplot(x='Risk', kind='count', hue='Installment plans', data=df,palette='coolwarm')
```

Out[38]:

<seaborn.axisgrid.FacetGrid at 0x199b6546cd0>

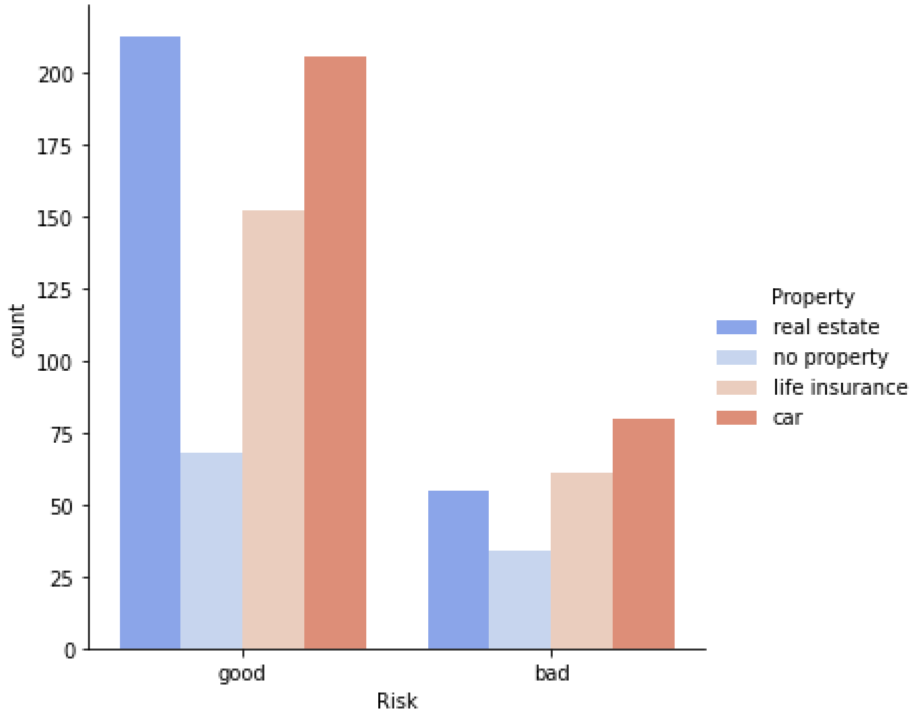


In [39]:

```
sns.factorplot(x='Risk', kind='count', hue='Property', data=df,palette='coolwarm')
```

Out[39]:

<seaborn.axisgrid.FacetGrid at 0x199b65c4100>

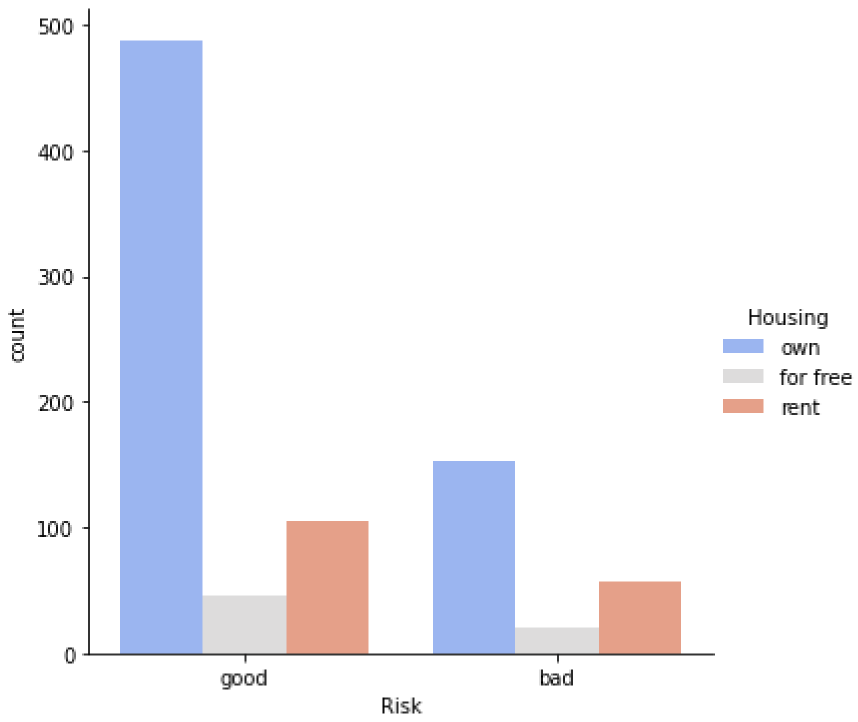


In [40]:

```
sns.factorplot(x='Risk', kind='count', hue='Housing', data=df,palette='coolwarm')
```

Out[40]:

<seaborn.axisgrid.FacetGrid at 0x199b4d34970>

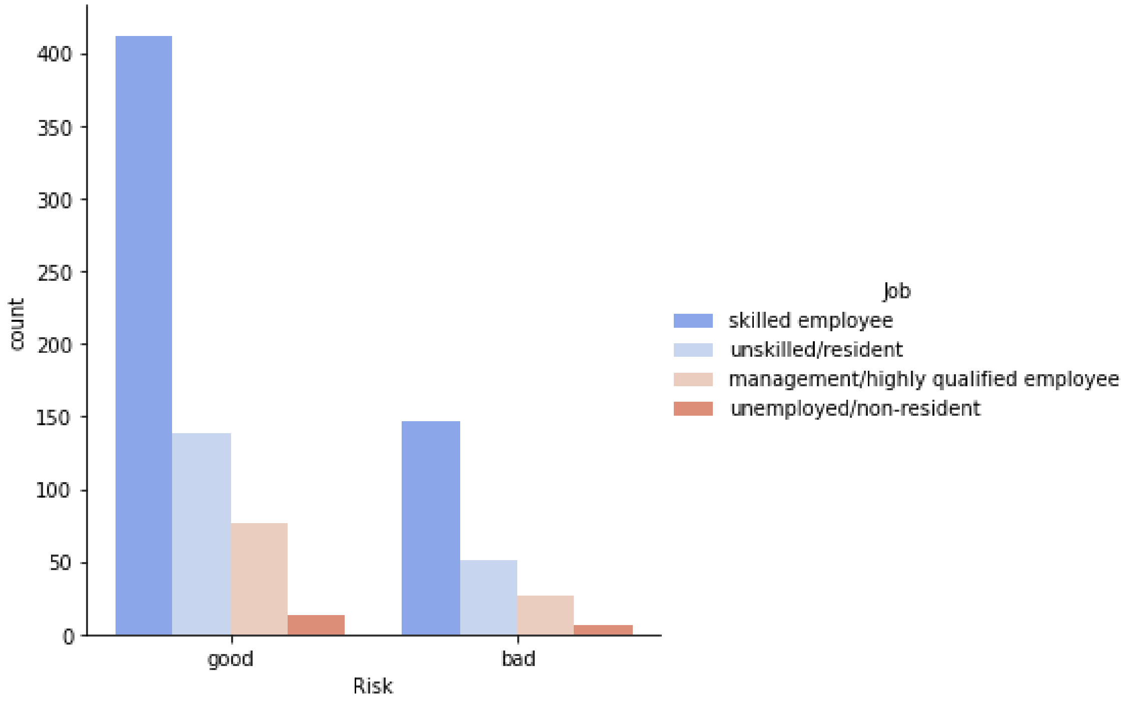


In [41]:

```python
sns.factorplot(x='Risk', kind='count', hue='Job', data=df,palette='coolwarm')
```

Out[41]:

<seaborn.axisgrid.FacetGrid at 0x199b7651a60>



In [42]:

```python
n_features = ['Duration', 'Installment Rate', 'Existing Credits', 'Age']
n_df = df[n_features + ['Risk']]
t_df = n_df
t_df.head()
```

Out[42]:

	Duration	Installment Rate	Existing Credits	Age	Risk
0	6	4	2	67	good
2	12	2	1	49	good
4	24	3	2	53	bad
6	24	3	1	53	good
7	36	2	1	35	good

3

In [43]:

```python
X = t_df.drop('Risk', axis=1)
Y = t_df['Risk']
```

In [44]:

```
from sklearn.model_selection import train_test_split
```

In [45]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

In [46]:

```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
print(0.8 * n_df.shape[0])
print(0.2 * n_df.shape[0])
```

```
(695, 4)
(174, 4)
(695,)
(174,)
695.2
173.8
```

In [47]:

```
from sklearn.tree import DecisionTreeClassifier
model1 = DecisionTreeClassifier(max_depth=3)
model1.fit(X_train,Y_train)
```

Out[47]:

```
DecisionTreeClassifier(max_depth=3)
```

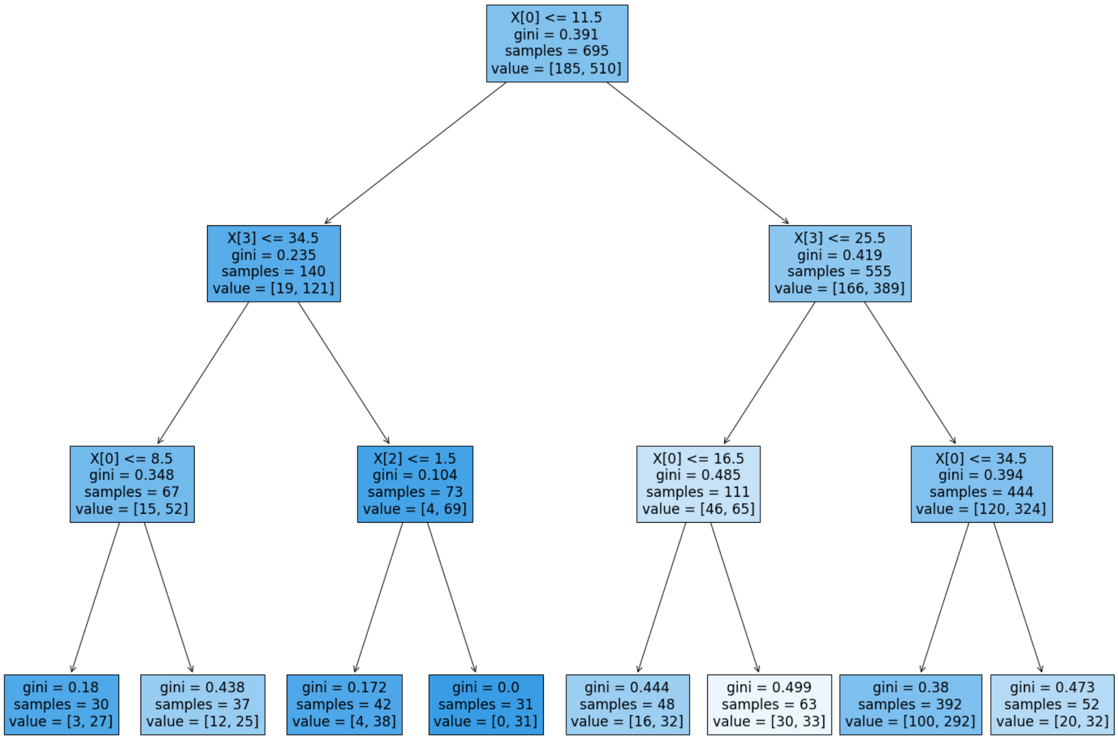
In [ ]:

```

```

In [48]:

```
from sklearn import tree
fig =plt.figure(figsize=(25,20))
_ = tree.plot_tree(model1, filled=True)
```



In [49]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, Y_train)
```

Out[49]:

```
RandomForestClassifier()
```

In [50]:

```
from sklearn.model_selection import train_test_split
```

In [51]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

In [52]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, Y_train)
```

Out[52]:

```
RandomForestClassifier()
```

In [53]:

```
rfc_pred = rfc.predict(X_test)
print(classification_report(Y_test, rfc_pred))
print(accuracy_score(Y_test, rfc_pred))
```

```
              precision    recall  f1-score   support

         bad       0.42      0.29      0.34        45
        good       0.78      0.86      0.82       129

    accuracy                           0.71       174
   macro avg       0.60      0.57      0.58       174
weighted avg       0.68      0.71      0.69       174

0.7126436781609196
```

4

In [54]:

```
S_A = pd.get_dummies(df['Saving Account'],drop_first=True)
S_A
```

Out[54]:

	moderate	quite rich	rich	unknown
0	0	0	0	1
2	0	0	0	0
4	0	0	0	0
6	0	1	0	0
7	0	0	0	0
...	...	...	...	...
995	0	0	0	0
996	0	0	0	1
997	0	0	0	0
998	0	0	0	0

In [55]:

```
C_A = pd.get_dummies(df['Checking Account'],drop_first=True)
C_A
```

Out[55]:

	little	moderate	rich
0	1	0	0
2	0	0	0
4	1	0	0
6	0	0	0
7	0	1	0
...	...	...	...
995	1	0	0
996	0	0	0
997	0	0	0
998	1	0	0
999	0	0	0

869 rows × 3 columns

In [56]:

```
E_S = pd.get_dummies(df['Employment Status'],drop_first=True)
E_S
```

Out[56]:

	4 to <7 years	< 1 year	>= 7 years	unemployed
0	0	0	1	0
2	1	0	0	0
4	0	0	0	0
6	0	0	1	0
7	0	0	0	0
...	...	...	...	...
995	0	0	0	1
996	0	0	1	0
997	1	0	0	0
998	0	0	0	0
999	0	0	1	0

869 rows × 4 columns

In [57]:

```
P = pd.get_dummies(df['Purpose'],drop_first=True)
P
```

Out[57]:

	car	domestic appliances	education	furniture/equipment	others	radio/television	repairs	retraini
0	0	0	0	0	0	1	0	
2	0	0	1	0	0	0	0	
4	1	0	0	0	0	0	0	
6	0	0	0	1	0	0	0	
7	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	
995	0	0	0	1	0	0	0	
996	1	0	0	0	0	0	0	
997	0	0	0	1	0	0	0	
998	1	0	0	0	0	0	0	
999	0	0	0	0	0	1	0	

869 rows × 8 columns

In [58]:

```
P_S = pd.get_dummies(df['Personal Status'],drop_first=True)
P_S
```

Out[58]:

	male : divorced/separated	male : married/widowed	male : single
0	0	0	1
2	0	0	1
4	0	0	1
6	0	0	1
7	0	0	1
...	...	...	...
995	0	0	1
996	0	0	1
997	0	0	0
998	1	0	0
999	0	0	1

869 rows × 3 columns

In [59]:

```
Pt = pd.get_dummies(df['Property'],drop_first=True)
Pt
```

Out[59]:

	life insurance	no property	real estate
0	0	0	1
2	0	0	1
4	0	1	0
6	1	0	0
7	0	0	0
...	...	...	...
995	1	0	0
996	0	0	0
997	0	0	1
998	1	0	0
999	0	0	0

869 rows × 3 columns

In [60]:

```
Guarantors = pd.get_dummies(df['Guarantors'],drop_first=True)
Guarantors
```

Out[60]:

	guarantor	none
0	0	1
2	0	1
4	0	1
6	0	1
7	0	1
...	...	...
995	0	1
996	0	1
997	0	1
998	0	1
999	0	1

869 rows × 2 columns

In [61]:

```
I_p = pd.get_dummies(df['Installment plans'],drop_first=True)
I_p
```

Out[61]:

	none	stores
0	1	0
2	1	0
4	1	0
6	1	0
7	1	0
...	...	...
995	1	0
996	1	0
997	1	0
998	1	0
999	1	0

869 rows × 2 columns

In [62]:

```
Job = pd.get_dummies(df['Job'],drop_first=True)
Job
```

Out[62]:

	skilled employee	unemployed/non-resident	unskilled/resident
0	1	0	0
2	0	0	1
4	1	0	0
6	1	0	0
7	0	0	0
...	...	...	...
995	0	0	0
996	1	0	0
997	0	0	1
998	0	0	0
999	1	0	0

869 rows × 3 columns

In [63]:

```
Housing = pd.get_dummies(df['Housing'],drop_first=True)
Housing
```

Out[63]:

	own	rent
0	1	0
2	1	0
4	0	0
6	1	0
7	0	1
...	...	...
995	1	0
996	1	0
997	1	0
998	1	0
999	1	0

869 rows × 2 columns

In [ ]:

```

```

In [ ]:

```

```

In [64]:

```
train_df = pd.concat([t_df, Guarantors, Pt, P_S,I_p], axis=1)
```

In [67]:

```
train_df.head()

```

Out[67]:

	Duration	Installment Rate	Existing Credits	Age	Risk	guarantor	none	life insurance	no property	real estate
0	6	4	2	67	good	0	1	0	0	1
2	12	2	1	49	good	0	1	0	0	1
4	24	3	2	53	bad	0	1	0	1	0
6	24	3	1	53	good	0	1	1	0	0
7	36	2	1	35	good	0	1	0	0	0

In [91]:

```
P_S
```

Out[91]:

	male : divorced/separated	male : married/widowed	male : single
0	0	0	1
2	0	0	1
4	0	0	1
6	0	0	1
7	0	0	1
...	...	...	...
995	0	0	1
996	0	0	1
997	0	0	0
998	1	0	0
999	0	0	1

869 rows × 3 columns

In [92]:

```
X = t_df.drop('Risk',axis=1)
Y = t_df['Risk']
```

In [93]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

In [94]:

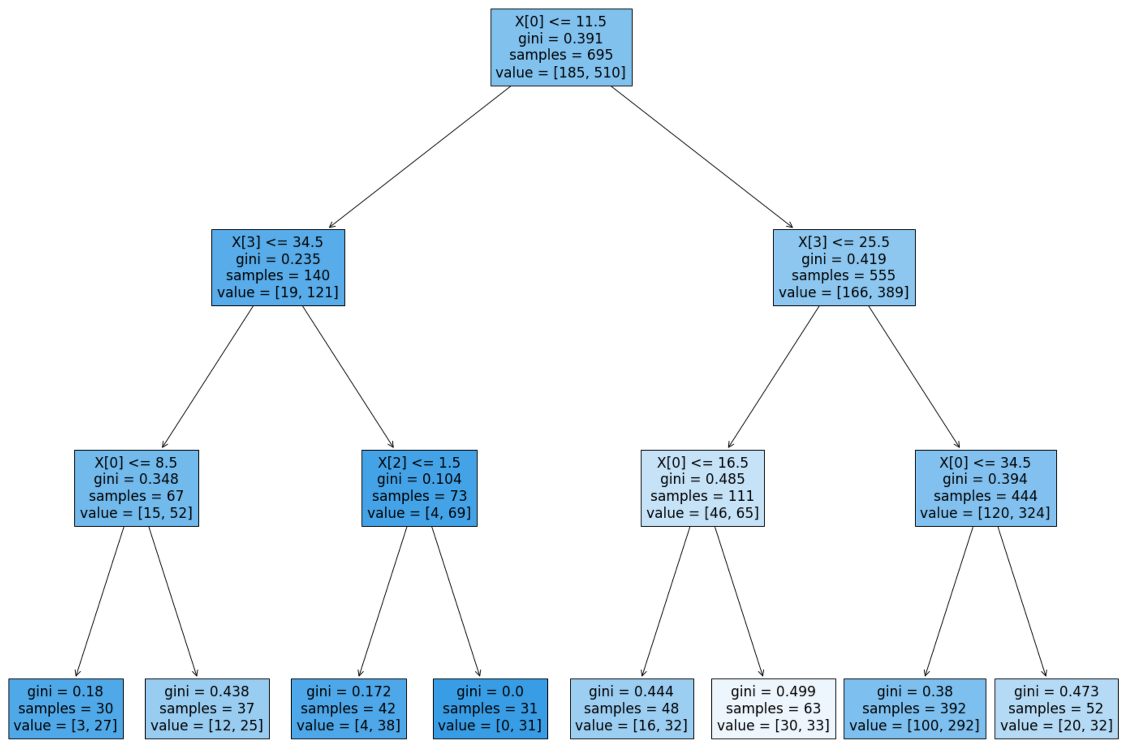
```
from sklearn.tree import DecisionTreeClassifier
model4 = DecisionTreeClassifier(max_depth=3)
model4.fit(X_train,Y_train)
```

Out[94]:

```
DecisionTreeClassifier(max_depth=3)
```

In [95]:

```
from sklearn import tree
fig =plt.figure(figsize=(25,20))
_ = tree.plot_tree(model4, filled=True)
```



In [96]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, Y_train)
```

Out[96]:

```
RandomForestClassifier()
```

In [97]:

```
from sklearn.model_selection import train_test_split
```

In [98]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

In [99]:

```
rfc_pred = rfc.predict(X_test)
print(classification_report(Y_test, rfc_pred))
print(accuracy_score(Y_test, rfc_pred))
```

```
              precision    recall  f1-score   support

         bad       0.43      0.27      0.33        45
        good       0.77      0.88      0.82       129

    accuracy                           0.72       174
   macro avg       0.60      0.57      0.58       174
weighted avg       0.68      0.72      0.69       174

0.7183908045977011
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```