HW3

September 29, 2025

1 K Nearest Neighbors Exercise

Maximum Number of points: 30

Questions 1-15 (2 points per question).

Import usual libraries.

[148]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

1. Read the knn_exercise_data.csv file into a dataframe

[149]:
```

```

2. Check the head of the dataframe.

[150]:
```

```

[150]:
```
           XVPM         GWYH         TRAT        TLLZ         IGGA  \
0  1636.670614   817.988525  2565.995189  358.347163   550.417491
1  1013.402760   577.587332  2644.141273  280.428203  1161.873391
2  1300.035501   820.518697  2025.854469  525.562292   922.206261
3  1059.347542  1066.866418   612.000041  480.827789   419.467495
4  1018.340526  1313.679056   950.622661  724.742174   843.065903

          HYKR         EDFS        GUUB         MGJM         JHZC  \
0  1618.870897  2147.641254  330.727893  1494.878631   845.136088
1  2084.107872   853.404981  447.157619  1193.032521   861.081809
2  2552.355407   818.676686  845.491492  1968.367513  1647.186291
3   685.666983   852.867810  341.664784  1154.391368  1450.935357
4  1370.554164   905.469453  658.118202   539.459350  1899.850792

   TARGET CLASS
0             0
```

1	1
2	1
3	0
4	0

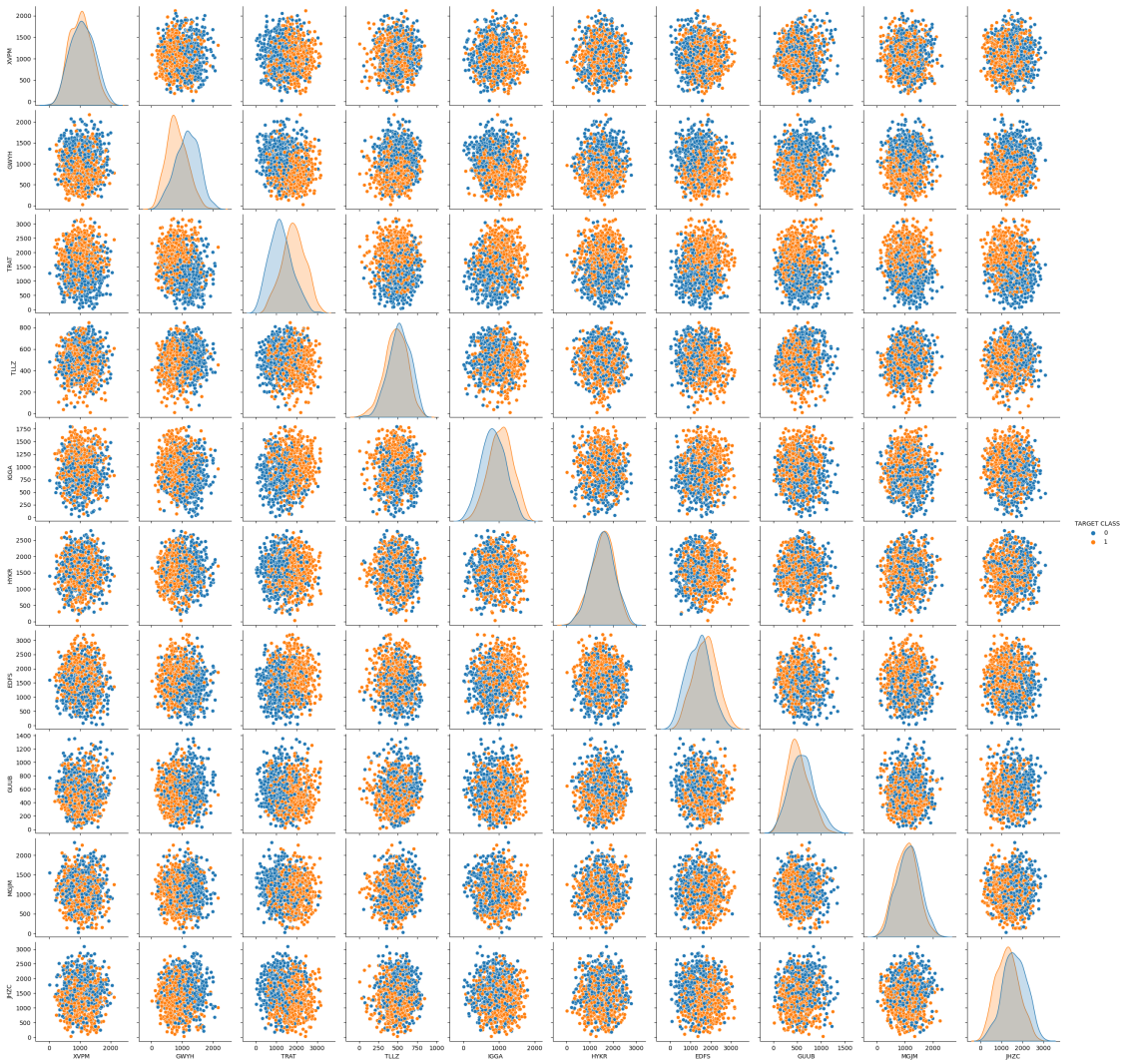
2 Explore the data

3. Use seaborn to create a pairplot with the hue indicated by the 'TARGET CLASS' column.

This will be a large plot … it may take a while

[151]:

[151]: <seaborn.axisgrid.PairGrid at 0x28457435a90>



3 Standardize the Feature Variables

4. Use StandardScaler from Scikit learn to scale the features. * Create a StandardScaler() object called scaler. * Fit scaler to the features. * Transform the features to a scaled version.

[152]:
```
from sklearn.preprocessing import StandardScaler
```

[153]:
```

```

[154]:
```

```

[154]:
```
StandardScaler()
```

[ ]:
```
scaled_features = ### FILL IN YOUR CODE HERE ###
```

Convert the scaled features to a dataframe.

[ ]:
```
df_feat = pd.DataFrame(scaled_features,columns=df.columns[:-1])
```

Check the head of the dataframe to make sure the scaling worked.

[ ]:
```
df_feat.head()
```

[ ]:
```
       XVPM      GWYH      TRAT      TLLZ      IGGA      HYKR      EDFS  \
0  1.568522 -0.443435  1.619808 -0.958255 -1.128481  0.138336  0.980493
1 -0.112376 -1.056574  1.741918 -1.504220  0.640009  1.081552 -1.182663
2  0.660647 -0.436981  0.775793  0.213394 -0.053171  2.030872 -1.240707
3  0.011533  0.191324 -1.433473 -0.100053 -1.507223 -1.753632 -1.183561
4 -0.099059  0.820815 -0.904346  1.609015 -0.282065 -0.365099 -1.095644

       GUUB      MGJM      JHZC
0 -0.932794  1.008313 -1.069627
1 -0.461864  0.258321 -1.041546
2  1.149298  2.184784  0.342811
3 -0.888557  0.162310 -0.002793
4  0.391419 -1.365603  0.787762
```

3.1 Training set and testing set.

5. Split the data into a training set and a testing set using 'train_test_split' with 'random_state' set to 101 and test_size=0.3.

[158]:
```
from sklearn.model_selection import train_test_split
```

[159]:
```

```

3.2 Create and Fit KNN Model

[160]:
```
from sklearn.neighbors import KNeighborsClassifier
```

6. Create a KNN model instance with n_neighbors=1

[161]:
```

```

7. Fit the KNN model to the training data.

[162]:
```

```

[162]:
```
KNeighborsClassifier(n_neighbors=1)
```

3.3 Predict and Evaluate

8. Predict values using the KNN model and X_test.

[163]:
```

```

9-10. Create a confusion matrix and classification report.

[164]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

[165]:
```

```

```
[[109  43]
 [ 41 107]]
```

[166]:
```

```

	precision	recall	f1-score	support
0	0.73	0.72	0.72	152
1	0.71	0.72	0.72	148
accuracy			0.72	300
macro avg	0.72	0.72	0.72	300
weighted avg	0.72	0.72	0.72	300

4 Choose a K Value

11-12. Compute the error rate for all K from 1 to 40. Use the elbow method to pick a good K Value.

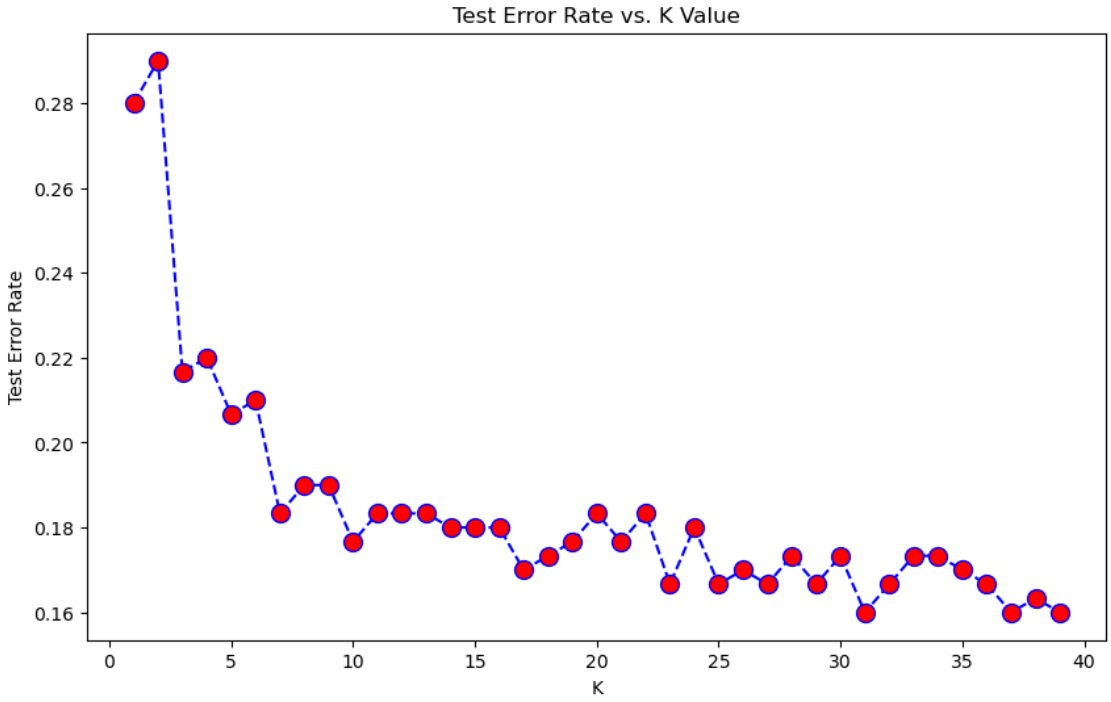
[167]:
```

```

[168]:
```

```

[168]:
```
Text(0, 0.5, 'Test Error Rate')
```



13-15. Retrain your model with the best K value, compute the test error rate for the best K value, and re-do the classification report and the confusion matrix.

[ ]:

[ ]:

[ ]:

[ ]:

5 DecisionTreeRandomForestExercise

Maximum Number of points: 40

Questions 1-20 (2 points per question)

Background Lending Club connects people who need money (borrowers) with people who have money (investors). An investor would want to invest in people who showed a profile of having a high probability of paying back.

You will use publicly available lending data from 2007-2010 (**loan_data.csv** file) to develop and mode to classify and predict whether or not the borrower paid back their loan in full. The file provided has been cleaned..

Here are what the columns represent: * credit.policy: 1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise. * purpose: The purpose of the loan (takes

values “credit_card”, “debt_consolidation”, “educational”, “major_purchase”, “small_business”, and “all_other”). * int.rate: The interest rate of the loan, as a proportion (a rate of 11% would be stored as 0.11). Borrowers judged by LendingClub.com to be more risky are assigned higher interest rates. * installment: The monthly installments owed by the borrower if the loan is funded. * log.annual.inc: The natural log of the self-reported annual income of the borrower. * dti: The debt-to-income ratio of the borrower (amount of debt divided by annual income). * fico: The FICO credit score of the borrower. * days.with.cr.line: The number of days the borrower has had a credit line. * revol.bal: The borrower’s revolving balance (amount unpaid at the end of the credit card billing cycle). * revol.util: The borrower’s revolving line utilization rate (the amount of the credit line used relative to total credit available). * inq.last.6mths: The borrower’s number of inquiries by creditors in the last 6 months. * delinq.2yrs: The number of times the borrower had been 30+ days past due on a payment in the past 2 years. * pub.rec: The borrower’s number of derogatory public records (bankruptcy filings, tax liens, or judgments).

6 Import Libraries

Import the usual libraries for pandas and plotting. You can import sklearn later on.

[103]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set_style("darkgrid")
import warnings
warnings.filterwarnings('ignore')
```

6.1 Get the Data

1. Use pandas to read loan_data.csv as a dataframe called loans.

[104]:
```

```

2. Check out the info(), head(), and describe() methods on loans.

[105]:
```

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   credit.policy      9578 non-null   int64
 1   purpose            9578 non-null   object
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   fico               9578 non-null   int64
```

```
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64
 11  delinq.2yrs        9578 non-null   int64
 12  pub.rec            9578 non-null   int64
 13  not.fully.paid     9578 non-null   int64
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

[106]:

[106]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	\
0	1	debt_consolidation	0.1189	829.10	11.350407	
1	1	credit_card	0.1071	228.22	11.082143	
2	1	debt_consolidation	0.1357	366.86	10.373491	
3	1	debt_consolidation	0.1008	162.34	11.350407	
4	1	credit_card	0.1426	102.92	11.299732	

	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	\
0	19.48	737	5639.958333	28854	52.1	0	
1	14.29	707	2760.000000	33623	76.7	0	
2	11.63	682	4710.000000	3511	25.6	1	
3	8.10	712	2699.958333	33667	73.2	1	
4	14.97	667	4066.000000	4740	39.5	0	

	delinq.2yrs	pub.rec	not.fully.paid
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	1	0	0

[107]:

[107]:

	credit.policy	int.rate	installment	log.annual.inc	dti	\
count	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	
mean	0.804970	0.122640	319.089413	10.932117	12.606679	
std	0.396245	0.026847	207.071301	0.614813	6.883970	
min	0.000000	0.060000	15.670000	7.547502	0.000000	
25%	1.000000	0.103900	163.770000	10.558414	7.212500	
50%	1.000000	0.122100	268.950000	10.928884	12.665000	
75%	1.000000	0.140700	432.762500	11.291293	17.950000	
max	1.000000	0.216400	940.140000	14.528354	29.960000	

	fico	days.with.cr.line	revol.bal	revol.util	\
count	9578.000000	9578.000000	9.578000e+03	9578.000000	
mean	710.846314	4560.767197	1.691396e+04	46.799236	

```
std      37.970537        2496.930377  3.375619e+04    29.014417
min     612.000000         178.958333  0.000000e+00     0.000000
25%     682.000000        2820.000000  3.187000e+03    22.600000
50%     707.000000        4139.958333  8.596000e+03    46.300000
75%     737.000000        5730.000000  1.824950e+04    70.900000
max     827.000000       17639.958330  1.207359e+06   119.000000

       inq.last.6mths  delinq.2yrs      pub.rec  not.fully.paid
count     9578.000000  9578.000000  9578.000000     9578.000000
mean         1.577469     0.163708     0.062122        0.160054
std          2.200245     0.546215     0.262126        0.366676
min          0.000000     0.000000     0.000000        0.000000
25%          0.000000     0.000000     0.000000        0.000000
50%          1.000000     0.000000     0.000000        0.000000
75%          2.000000     0.000000     0.000000        0.000000
max         33.000000    13.000000     5.000000        1.000000
```

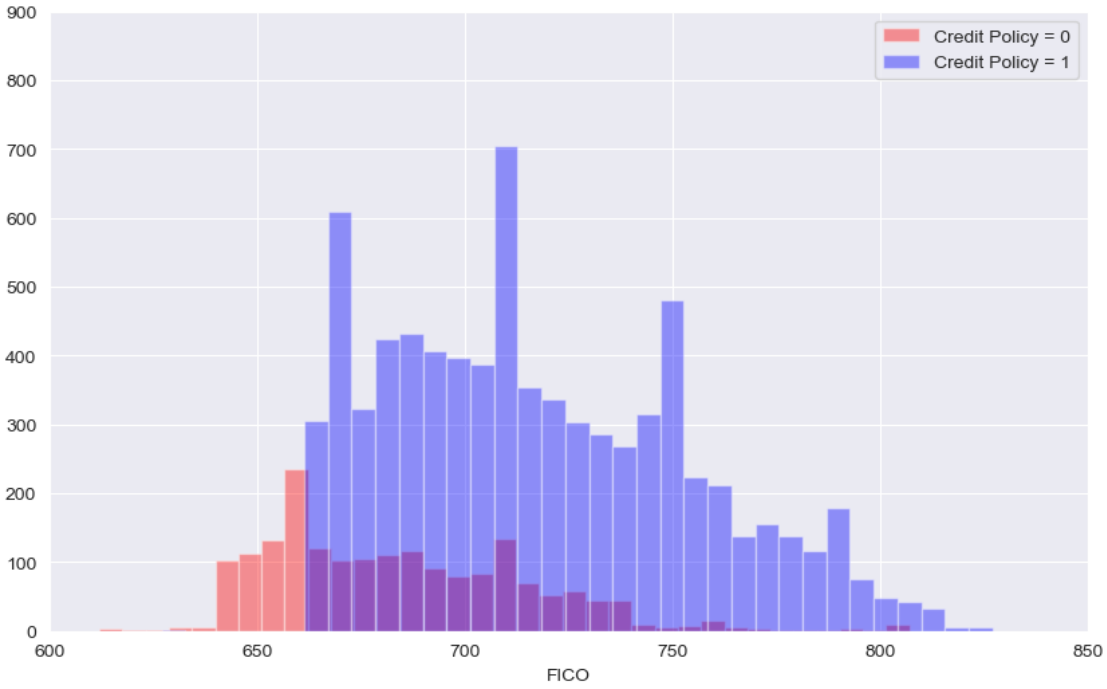
7 Exploratory Data Analysis

Use any matplotlib, seaborn, pandas built-in plotting libraries to create plots similar to those below. Don't worry about the colors matching, etc., just get the main idea of the plot.

3. Create a histogram of two FICO distributions on top of each other, one for each credit.policy outcome.

[108]:

[108]: Text(0.5, 0, 'FICO')



4. Create a similar figure, except this time select by the not.fully.paid column.

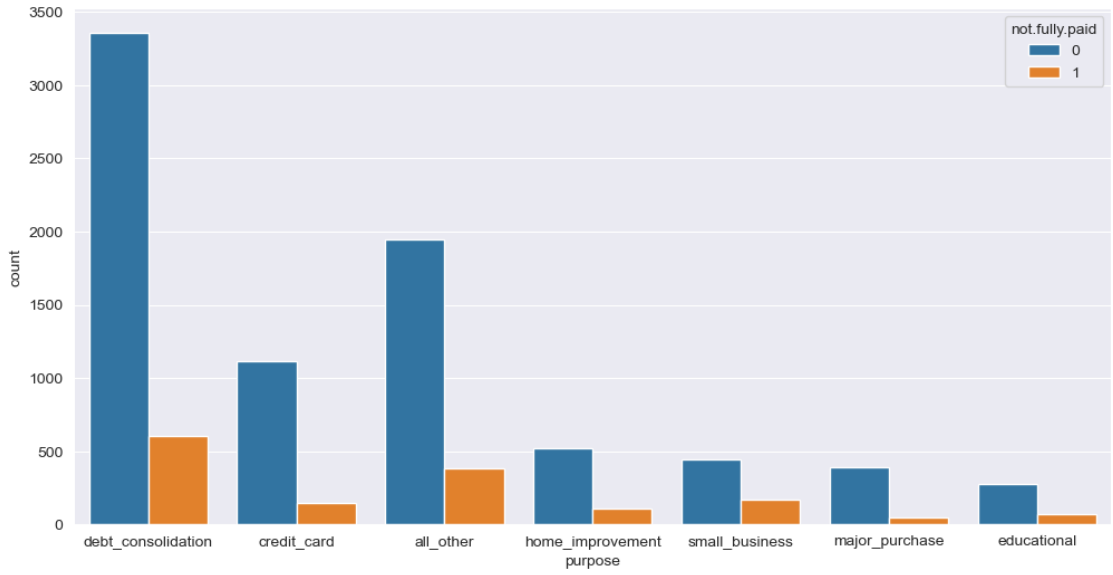
[109]:

[109]: Text(0.5, 0, 'FICO')

5. Create a countplot using seaborn showing the counts of loans by purpose, with the color hue defined by not.fully.paid.

[110]:

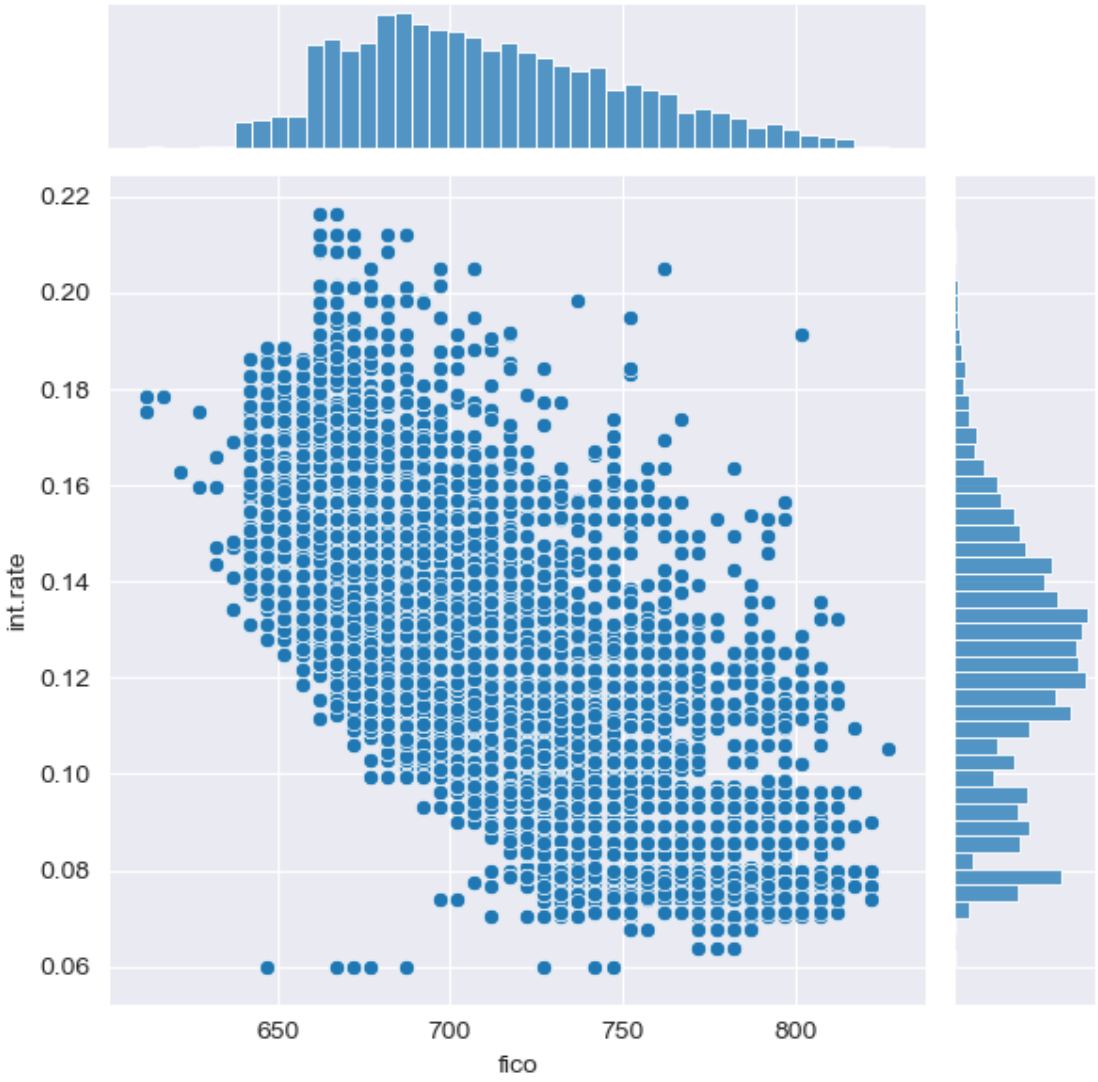
[110]: <AxesSubplot:xlabel='purpose', ylabel='count'>



6. Create jointplot showing the trend between FICO score and interest rate.

[111]:

[111]: <seaborn.axisgrid.JointGrid at 0x257d6c2a550>



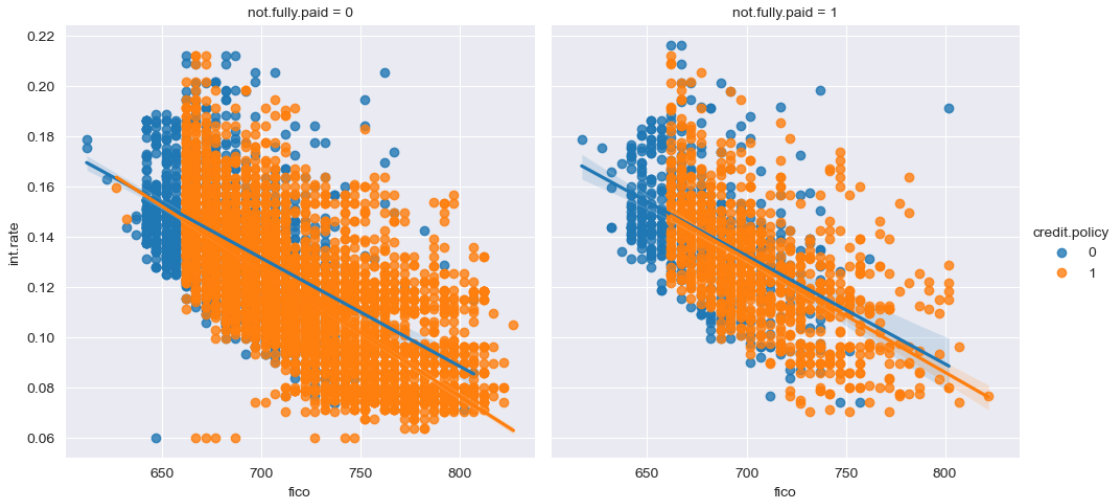
The plot above shows that 'int.rate' and 'fico' are inversely related. This means that people with higher FICO score are less risky and so they are assigned smaller interest rates.

7. Create the following lmplots to see if the trend differed between not.fully.paid and credit.policy.

Check the documentation for lmplot() (look for **col** parameter) if you can't figure out how to separate it into columns.**

[112]:

[112]: `<seaborn.axisgrid.FacetGrid at 0x257d6c1e250>`



7.1 Set up the Data

Notice (from loans.info() earlier) that the **purpose** column is categorical.
Need to transform it for sklearn to understand. Use dummy variables.

Create a list of 1 element containing the string 'purpose'. Call this list cat_feats.

[113]:
```
cat_feats = ['purpose']
```

Now use pd.get_dummies(loans,columns=cat_feats,drop_first=True) to create a fixed larger dataframe that has new feature columns with dummy variables. Set this dataframe as final_data. (don't forget to check out the get_dummies documentation)

[114]:
```
final_data = pd.get_dummies(loans, columns = cat_feats, drop_first = True)
final_data.head()
```

[114]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	\
0	1	0.1189	829.10	11.350407	19.48	737	
1	1	0.1071	228.22	11.082143	14.29	707	
2	1	0.1357	366.86	10.373491	11.63	682	
3	1	0.1008	162.34	11.350407	8.10	712	
4	1	0.1426	102.92	11.299732	14.97	667	

	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	\
0	5639.958333	28854	52.1	0	0	
1	2760.000000	33623	76.7	0	0	
2	4710.000000	3511	25.6	1	0	
3	2699.958333	33667	73.2	1	0	
4	4066.000000	4740	39.5	0	1	

	pub.rec	not.fully.paid	purpose_credit_card	purpose_debt_consolidation	\

```
0        0               0                     0                           1
1        0               0                     1                           0
2        0               0                     0                           1
3        0               0                     0                           1
4        0               0                     1                           0

   purpose_educational  purpose_home_improvement  purpose_major_purchase  \
0                    0                         0                       0
1                    0                         0                       0
2                    0                         0                       0
3                    0                         0                       0
4                    0                         0                       0

   purpose_small_business
0                       0
1                       0
2                       0
3                       0
4                       0
```

8. Display final_data info

[115]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 19 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   credit.policy               9578 non-null   int64
 1   int.rate                    9578 non-null   float64
 2   installment                 9578 non-null   float64
 3   log.annual.inc              9578 non-null   float64
 4   dti                         9578 non-null   float64
 5   fico                        9578 non-null   int64
 6   days.with.cr.line           9578 non-null   float64
 7   revol.bal                   9578 non-null   int64
 8   revol.util                  9578 non-null   float64
 9   inq.last.6mths              9578 non-null   int64
 10  delinq.2yrs                 9578 non-null   int64
 11  pub.rec                     9578 non-null   int64
 12  not.fully.paid              9578 non-null   int64
 13  purpose_credit_card         9578 non-null   uint8
 14  purpose_debt_consolidation  9578 non-null   uint8
 15  purpose_educational         9578 non-null   uint8
 16  purpose_home_improvement    9578 non-null   uint8
 17  purpose_major_purchase      9578 non-null   uint8
 18  purpose_small_business      9578 non-null   uint8
```

```
dtypes: float64(6), int64(7), uint8(6)
memory usage: 1.0 MB
```

7.2 Train Test Split

9. Import train_test_split and split the data (use 30% test size and random_state 101).

[116]:
```
from sklearn.model_selection import train_test_split
```

[117]:
```

```

7.3 Train a single Decision Tree Model

Import DecisionTreeClassifier

[118]:
```
from sklearn.tree import DecisionTreeClassifier
```

10. Create an instance of DecisionTreeClassifier() called dtree and fit it to the training data.

[119]:
```

```

[120]:
```

```

[120]:
```
DecisionTreeClassifier()
```

7.4 Predictions and Evaluation of Decision Tree

11-12. Create predictions from the test set and create a classification report and a confusion matrix.

[121]:
```

```

[122]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

[123]:
```

```

```
              precision    recall  f1-score   support

           0       0.85      0.82      0.84      2431
           1       0.19      0.23      0.21       443

    accuracy                           0.73      2874
   macro avg       0.52      0.52      0.52      2874
weighted avg       0.75      0.73      0.74      2874
```

[124]:
```

```

```
[[1987  444]
 [ 341  102]]
```

7.5 Train a Random Forest model

From sklearn.ensemble import RandomForestClassifier

[125]:
```
from sklearn.ensemble import RandomForestClassifier
```

13. Create an instance of the RandomForestClassifier class and fit it to the training data.

[126]:
```

```

[127]:
```

```

[127]:
```
RandomForestClassifier(n_estimators=300)
```

7.6 Predictions and Evaluation

14. Predict the class of not.fully.paid for the X_test data.

[128]:
```

```

15. Create a classification report from the results.

[129]:
```

```

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92      2431
           1       0.47      0.02      0.03       443

    accuracy                           0.85      2874
   macro avg       0.66      0.51      0.47      2874
weighted avg       0.79      0.85      0.78      2874
```

16. Show the Confusion Matrix for the predictions.

[130]:
```

```

```
[[2423    8]
 [ 436    7]]
```

17-20. Can we definitively conclude that the random forest outperformed the decision tree? Explain your answer.

[131]:
```
# Write your answer here.
```

7.7 HW4 - Support Vector Machines Exercise

Maximum Number of points: 30

Question 1-15 (2 points per question)

This exercise will use a famous data set, the "Iris flower data set".

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Sir Ronald Fisher in the 1936
The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor), so 150 total samples. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

A brief history of the data set as well as pictures of the Irises can be found here Iris flower data set.

The iris dataset contains measurements for 150 iris flowers from three different species.

The three classes in the Iris dataset:

```
Iris-setosa (n=50)
Iris-versicolor (n=50)
Iris-virginica (n=50)
```

The four features of the Iris dataset:

```
sepal length in cm
sepal width in cm
petal length in cm
petal width in cm
```

7.8 Get the data

Use seaborn to get the iris data by using: iris = sns.load_dataset('iris')

[29]:
```
import seaborn as sns
```

[30]:
```
iris = sns.load_dataset('iris')
```

7.9 Exploratory Data Analysis

Import some libraries you will need.

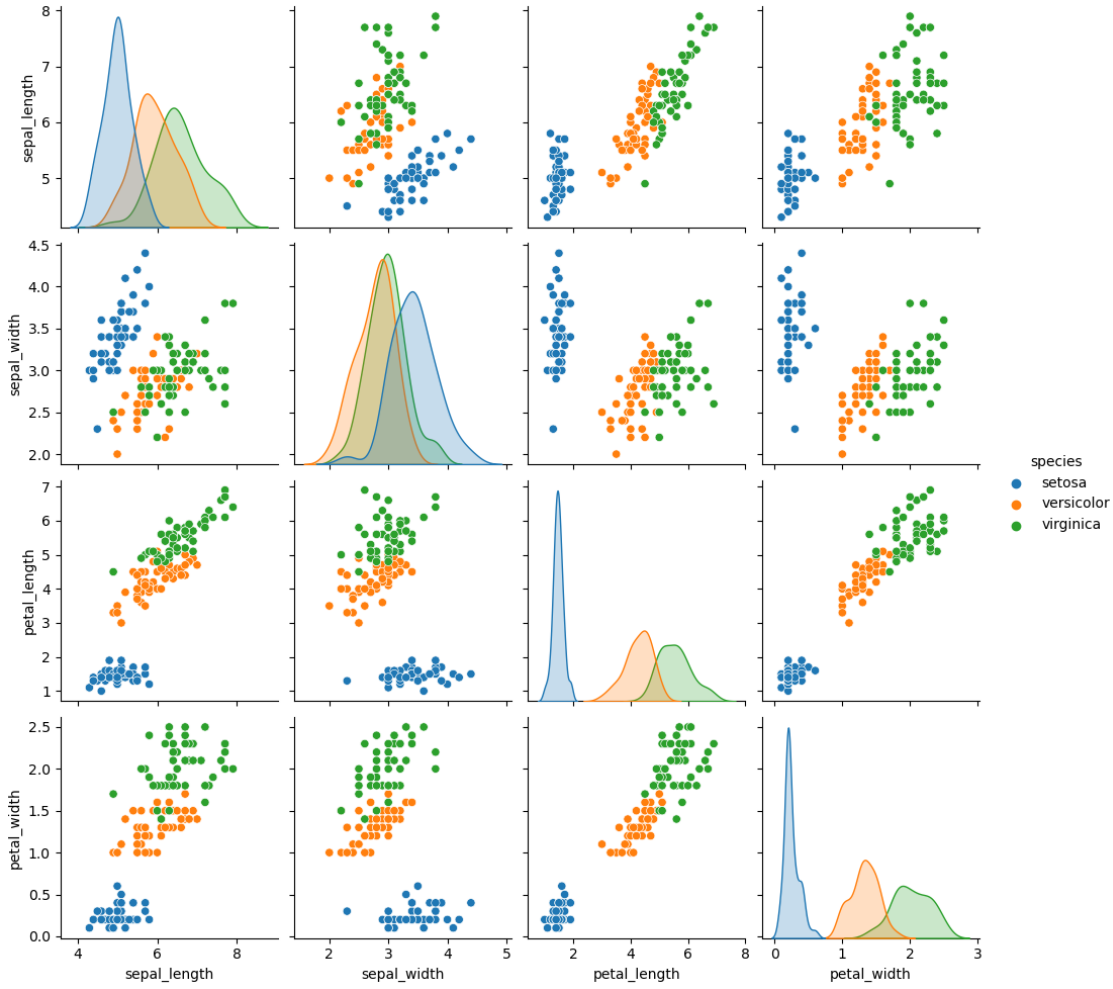
[31]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

1. Create a pairplot of the data set.

[32]:
```

```

[32]:
```
<seaborn.axisgrid.PairGrid at 0x1d56e852190>
```



2. Which flower species seems to be the most linearly separable?

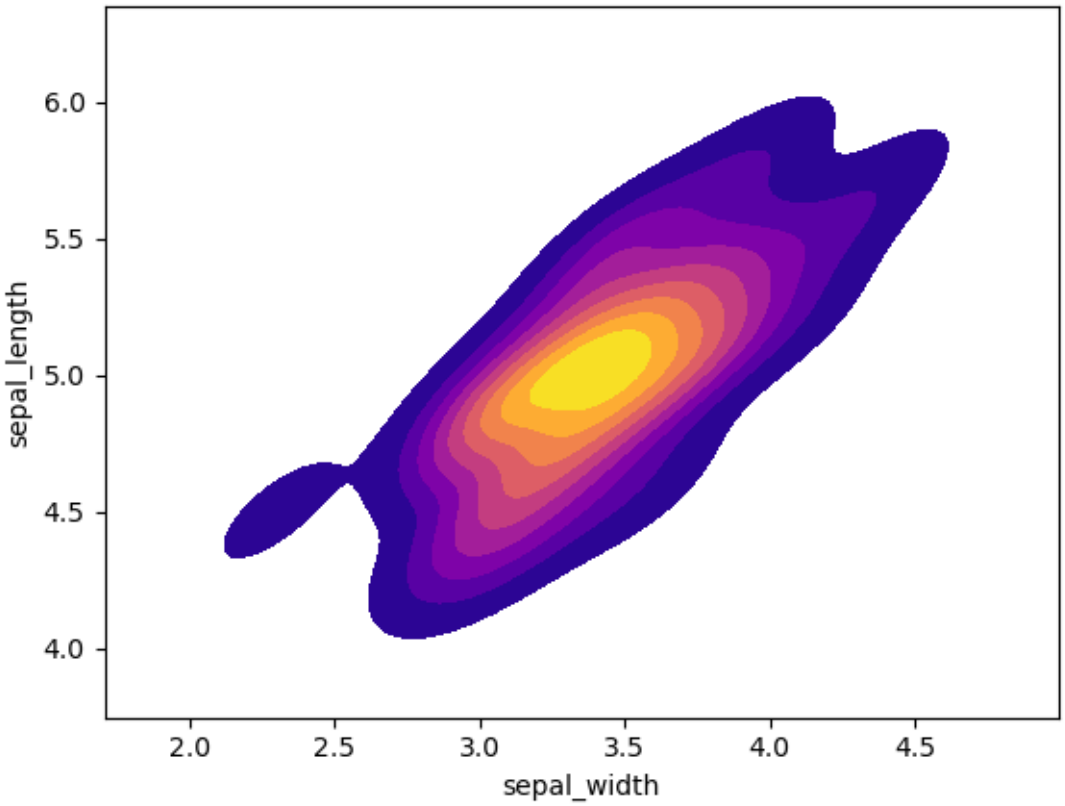
[33]:
```
# Write here your answer
```

3. Create a kde plot of sepal_length versus sepal width for setosa species of flower.

[34]:
```

```

[34]:
```
<AxesSubplot:xlabel='sepal_width', ylabel='sepal_length'>
```



7.10 Train Test Split

4. Split your data into a training set and a testing set.
Use test_size=0.30 and random_state=0

[35]:
```
from sklearn.model_selection import train_test_split
```

[36]:
```
```

7.11 Train a Support Vector Machine Classifier Model

5. Use the SVC() model from sklearn and fit the model.

[37]:
```
from sklearn.svm import SVC
```

[38]:
```
```

[39]:
```
```

[39]:
```
SVC()
```

7.12 Model Evaluation

6-7. Create a confusion matrix and a classification report.

[40]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

[41]:
```

```

[42]:
```

```

```
[[16  0  0]
 [ 0 17  1]
 [ 0  0 11]]
```

[43]:
```

```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        16
  versicolor       1.00      0.94      0.97        18
   virginica       0.92      1.00      0.96        11

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

You probably have noticed that this model was pretty good!
See if you can tune the parameters to try to do better (this is unlikely, and you probably would be satisfied with these results in real life because the data set is quite small, but I just want you to practice using GridSearch).

7.13 Gridsearch

Import GridsearchCV from SciKit Learn.

[44]:
```
from sklearn.model_selection import GridSearchCV
```

8. Create a dictionary called param_grid and fill out some parameters for C and gamma.
Your choice for the search grid. You do not have to use the same region as in my solutions.

[45]:
```

```

9. Create a GridSearchCV object and fit it to the training data.

[ ]:
```

```

10. For your search values, what were the best parameters?

[ ]:
```

```

11-13. Now take that grid model and create some predictions using the test set and create classification reports and confusion matrices for them.

[48]:

[ ]:

[ ]:

14-15. How do you compare the results from GridsearchCV and prior results? Explain your answer.

[ ]: