**Module 16**

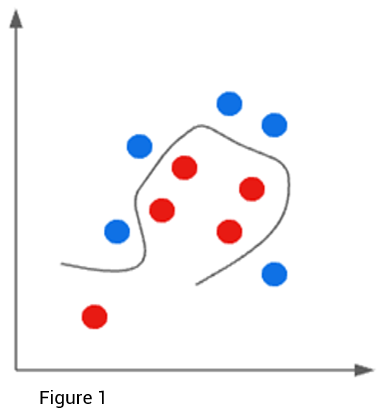
**Support Vector Machines (SVMs)**

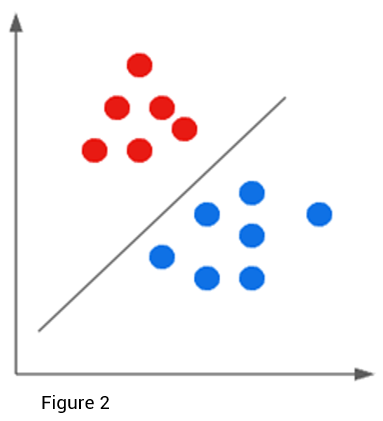
* [Video Transcripts](https://student.emeritus.org/courses/4765/files/3406592?wrap=1)
* [Download Video Transcripts](https://student.emeritus.org/courses/4765/files/3406592/download?download_frd=1)
* [Quick Reference Guide](https://student.emeritus.org/courses/4765/files/3406595?wrap=1)

**Notes:**

**The Kernel Trick**

In machine learning, a kernel refers to a method of solving a nonlinear problem, using a linear classifier. This involves separating linearly inseparable data (Figure 1) from linearly separable data (Figure 2). In each data instance, a kernel function maps the original nonlinear observations to a higher-dimensional space that can be separated.

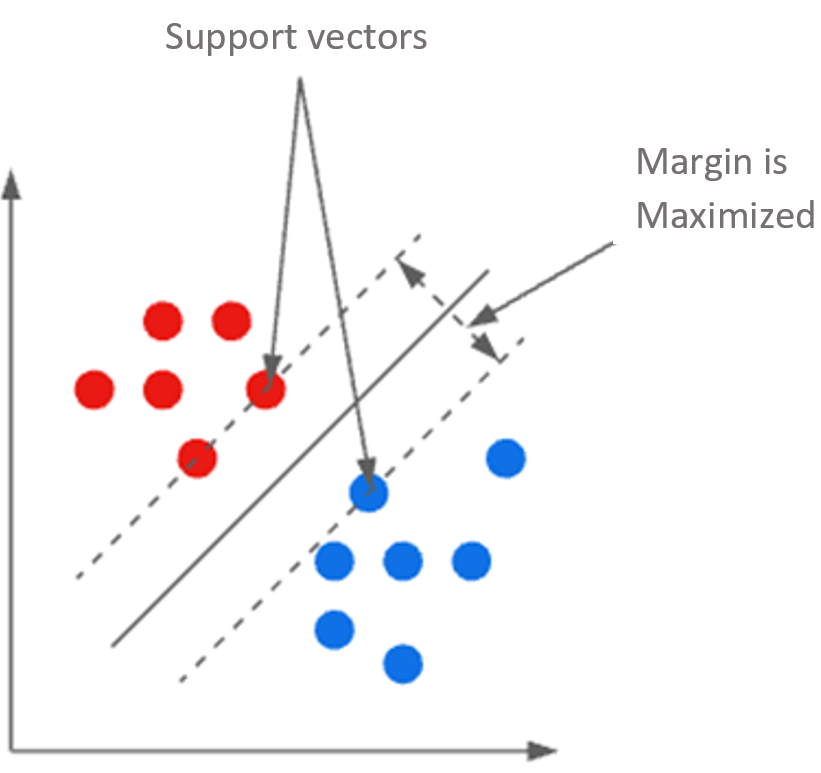




Now that you understand the kernel, the kernel trick will be easier to understand. The kernel trick allows you to find a decision surface that clearly differentiates between different classes if the data can be mapped from two-dimensional space to three-dimensional space. Then, as dimensional computations become more expensive within those spaces, the kernel can use the data in the original feature space without requiring coordinates in a higher-dimensional space.

**Maximum Margin Classifier**

An SVM creates two parallel hyperplanes that pass through the nearest data points. These nearest points are called support vectors, and the region between the support vectors is called the margin and is bounded by the two hyperplanes. These hyperplanes can be drawn in many different ways to classify the data. A hyperplane with the maximum margin is the most stable, and the margin indicates how far two classes are apart. In mathematics, the hyperplane with the highest margin is called the maximum margin hyperplane, and the classifier it defines is called the maximum margin classifier. Here is an illustration of the concept.



def make\_plot(estimator):

xx = np.linspace(X1.iloc[:, 0].min(), X1.iloc[:, 0].max(), 50)

yy = np.linspace(X1.iloc[:, 1].min(), X1.iloc[:, 1].max(), 50)

XX, YY = np.meshgrid(xx, yy)

grid = np.c\_[XX.ravel(), YY.ravel()]

labels = pd.factorize(estimator.predict(grid))[0]

plt.contourf(xx, yy, labels.reshape(XX.shape), cmap = 'twilight', alpha = 0.6)

sns.scatterplot(data = X1, x = 'total\_phenols', y = 'color\_intensity', hue = y, palette = 'flare')

**Module 16: Glossary**

**Hyperplane**

A flat subspace with dimension p − 1, given a p-dimensional space

**Kernel Trick**

A computational technique for enlarging the feature space

**Margin**

The distance between a hyperplane and the closest data point

**Maximum Margin Classifier**

A hyperplane whose margin is maximized

**Support Vectors**

Data points that are closer to the hyperplane and influence the position and orientation of the hyperplane

**Module Issues:**

**Codio Activity 16.1 Problem 1** ‘pipe’ is supposed to be ***lgr***

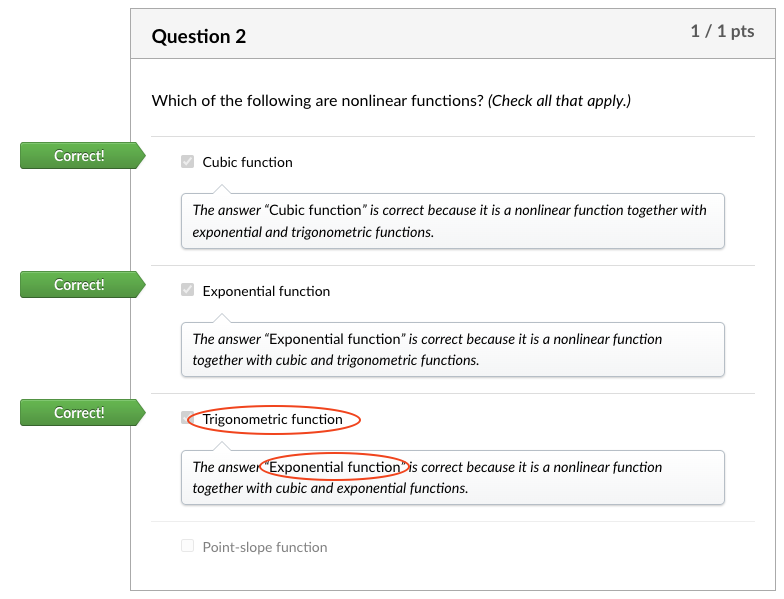
**Codio Activity 16.2 Problem 1 & 2** name 'scale' and 'model' in Pipeline for StandardScaler and KNeighborsClassifier respectively

**Codio Activity 16.2 Problem 2:** Make Variable **x to X** to make plot working!

* plt.plot(**x, lower(x)**, '--r')
* plt.plot(**x, upper(x)**, '--r')

**Codio Activity 16.9 Problem 1:** Make sure ***res\_dict*** has the dictionary of your dataframe because the hidden test is using it!

**Quiz Issue:**



**Quizes:**

Which of the following is not the quadratic feature generated from x0 and x1 linear features? : x0x0

*You are correct! The answer “*x0x0*” is correct because the quadratic features generated from x0 and x1 are “*x0x1*”, “*x02”*, and “*x12*”.*

Nonlinear features are used to indicate the curvature and to classify a small portion of the data points. : False

*You are correct! The answer “*False*” is correct because nonlinear features are used to indicate the curvature and to classify large portions of the data points.*

Which of the following are nonlinear functions? *(Check all that apply.) :* Exponential function, Cubic function, Trigonometric function

*The answer “*Exponential function*” is correct because it is a nonlinear function together with cubic and trigonometric functions.*

*The answer “*Cubic function*” is correct because it is a nonlinear function together with exponential and trigonometric functions.*

*The answer “*Exponential function*” is correct because it is a nonlinear function together with cubic and exponential functions.*

A kernel is a function that takes two data vectors as an input and returns a number. : True

*You are correct! The answer “*True*” is correct because the function of the kernel is to take data as input and transform it into the required form.*

What is the equation to determine the optimal β for linear regression with kernels? : β = (ΦTΦ + λIM)-1 ΦTY

*You are correct! The answer “*β = (ΦTΦ + λIM)-1 ΦTY*” is correct because this is the equation to determine the optimal β for linear regression with kernels.*

The equation shows the method for prediction with alphas.

y(xn)=∑i=1NαiϕT(xi)ϕ(xnew)

The prediction with alphas involves all the training data. : True

*You are correct! The answer “*True*” is correct because the equation has the term phi transpose(xi), which refers to involvement of the training data.*

The models built with a kernel-based approach have “n” coefficients (blank). : Alpha

*You are correct! The answer “*Alpha*” is correct because the models built with a kernel-based approach have “n” coefficients alpha.*

What is the code in Python to train a linear regression model given the “KernelMatrix”? : Linreg = sklearn.LinearRegression() Linreg.fit(KernelMatrix, train[‘Y’])

*You are correct! The answer “*Linreg = sklearn.LinearRegression() Linreg.fit(KernelMatrix, train[‘Y’])*” is correct because given the “KernelMatrix” this is the correct code in Python to train a linear regression model.*

What is the function in the Python library scikit-learn to build a Gaussian kernel function? : rbf\_kernel()

*You are correct! The answer “*rbf\_kernel()*” is correct because this is the function in Python library scikit-learn to build a gaussian kernel function.*

The maximum margin classifier is similar to logistic regression. : True

*You are correct! The answer “*True*” is correct because the maximum margin classifier, like logistic regression, produces linear boundaries and is amenable to the kernel trick.*

The margin of a decision boundary is the perpendicular distance from the boundary to the nearest data point in the training set. : True

*You are correct! The answer “*True*” is correct because the perpendicular distance from the boundary to the nearest data point in the training set is known as the margin of decision boundary.*

What is the generalized optimization problem for the maximum margin classifier? : Minimize ||β||2

*You are correct! The answer “*Minimize ||β||2*” is correct because minimizing the beta maximizes the margin, which is the best model.*

What do you call the data points that are used to determine the margin? : Support vectors

*You are correct! The answer “*Support vectors*” is correct because the data points that are used to determine the margin are known as support vectors.*

Using kernels for linear regression makes training more difficult when the kernel matrix is larger than the original dataset. : True

*You are correct! The answer “*True*” is correct because using kernels for linear regression makes training more difficult when the kernel matrix is larger than the original dataset.*

Which of the following are advantages of support vector machines? *(Check all that apply.)*

They have good numerical properties due to the relative sparsity of maximum margin classifiers

They are easy to construct in scikit-learn

They preserve the flexibility of kernels

*You are correct! The answer “*They have good numerical properties due to the relative sparsity of maximum margin classifiers*” is correct, as this is one of the advantages of support vector machines.*

*You are correct! The answer “*They are easy to construct in scikit-learn*” is correct, as this is one of the advantages of support vector machines.*

*You are correct! The answer “*They preserve the flexibility of kernels*” is correct, as this is one of the advantages of support vector machines.*

What is the package imported from the Python library “sklearn.svm” to use support vector machines? : SVC

*You are correct! The answer “*SVC*” is correct because this is the package imported from the Python library “*sklearn.svm*” to use support vector machines.*

Which of the following is not a constructor to the function “SVC()” when “(kernel = ’poly’)”? : Alpha

*You are correct! The answer “*Alpha*” is correct because this is not a constructor to the function “SVC()” when the kernel is set as “poly.”*

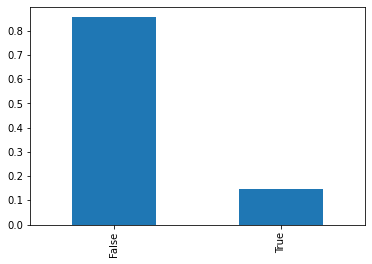
**Savio’s Session**

<http://www.tfidf.com/>

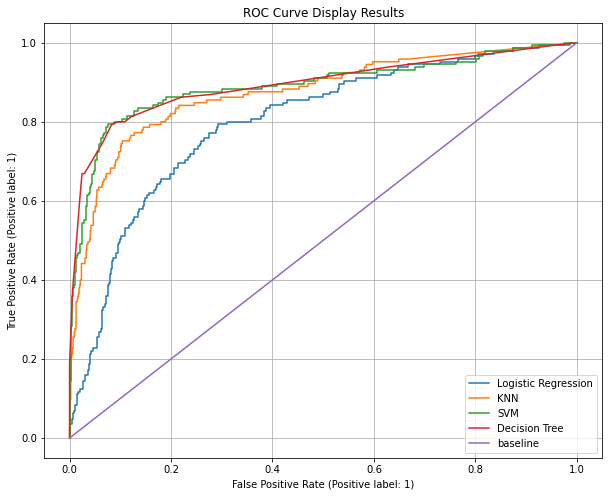
**Try-It Activity 16.1: Comparing Models - Section B**

This is a theoretical assignment, in general, imbalanced dataset is not a problem for KNN, however, Logistic Regression, Decision Trees and SVM do not perform well on imbalanced datasets.

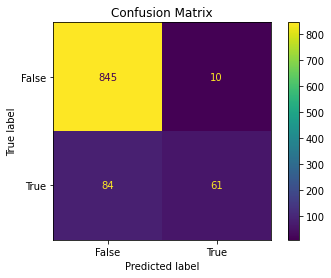
**Churn** dataset is **imbalanced**, it has 3333 entries and 20 columns:



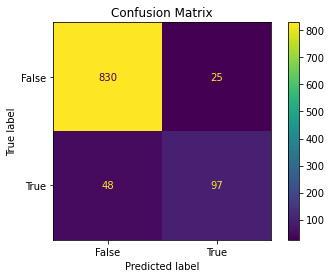
After scaling dataset, I got results from all 4 methods, ROC-AUC Curve Results show Decision Tree and SVM are pretty close, looking at their confusion matrix reveals Decision Tree (at 7 levels) performed more reliably in my case since it is binary classification I went with 'roc\_auc' in *scoring* parameter of GridSearchCV:



**SVM**



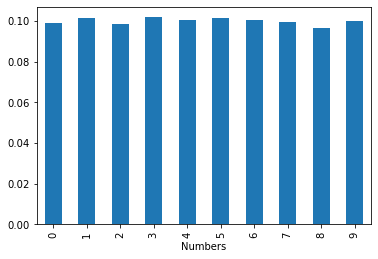
**Decision Tree**



**Churn Dataset Summary Table**

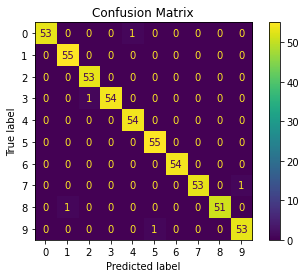
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Train Score | Test Score | Average Fit Time | Parameters | Is the model good at handling imbalanced classes? | Does the model train quickly? | Does the model yield interpretable results? |
| KNN | 1 | 0.873604 | 0.000652 | {'penalty': 'l1', 'solver': 'liblinear'} | Yes, KNN is not influenced by imbalanced dataset | Yes | Yes |
| Logistic Regression | 0.832071 | 0.798734 | 0.026334 | {'n\_neighbors': 23, 'weights': 'distance'} | No, It is not good at inbalanced classification | OK | Yes |
| SVC | 0.975603 | 0.889994 | 0.123106 | {'gamma': 0.1, 'kernel': 'rbf'} | OK | Slower | Yes |
| Decision Tree | 0.934038 | 0.894217 | 0.008184 | {'criterion': 'gini', 'max\_depth': 7, 'min\_samples\_leaf': 5, 'min\_samples\_split': 0.05} | OK | Yes | Yes |

**Digits** dataset is **balanced**, it has 1797 entries with 64 numeric array elements for digit representation:

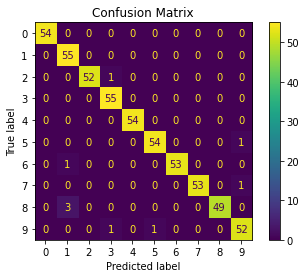


I ran all 4 methods directly with no manipulation of the dataset. SVM performed slightly better than KNN, a close look at their confusion matrix reveals, SVM has 5 misclassification versus 9 with KNN.

**SVM**



**KNN**



**Digits Dataset Summary Table**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Train Score | Test Score | Average Fit Time | Parameters | Is the model good at handling imbalanced classes? | Does the model train quickly? | Does the model yield interpretable results? |
| KNN | 1 | 0.983333 | 0.000443 | {'penalty': 'l1', 'solver': 'liblinear'} | Yes, KNN is not influenced by imbalanced dataset | Yes | Yes |
| Logistic Regression | 0.996818 | 0.961111 | 0.164151 | {'n\_neighbors': 1, 'weights': 'uniform'} | No, It is not good at imbalanced classification | Slower | Yes |
| SVC | 1 | 0.990741 | 0.057079 | {'gamma': 0.01, 'kernel': 'poly'} | OK | OK | Yes |
| Decision Tree | 0.845664 | 0.792593 | 0.006105 | {'criterion': 'entropy', 'max\_depth': 9, 'min\_samples\_leaf': 5, 'min\_samples\_split': 0.05} | OK | Yes | Yes |

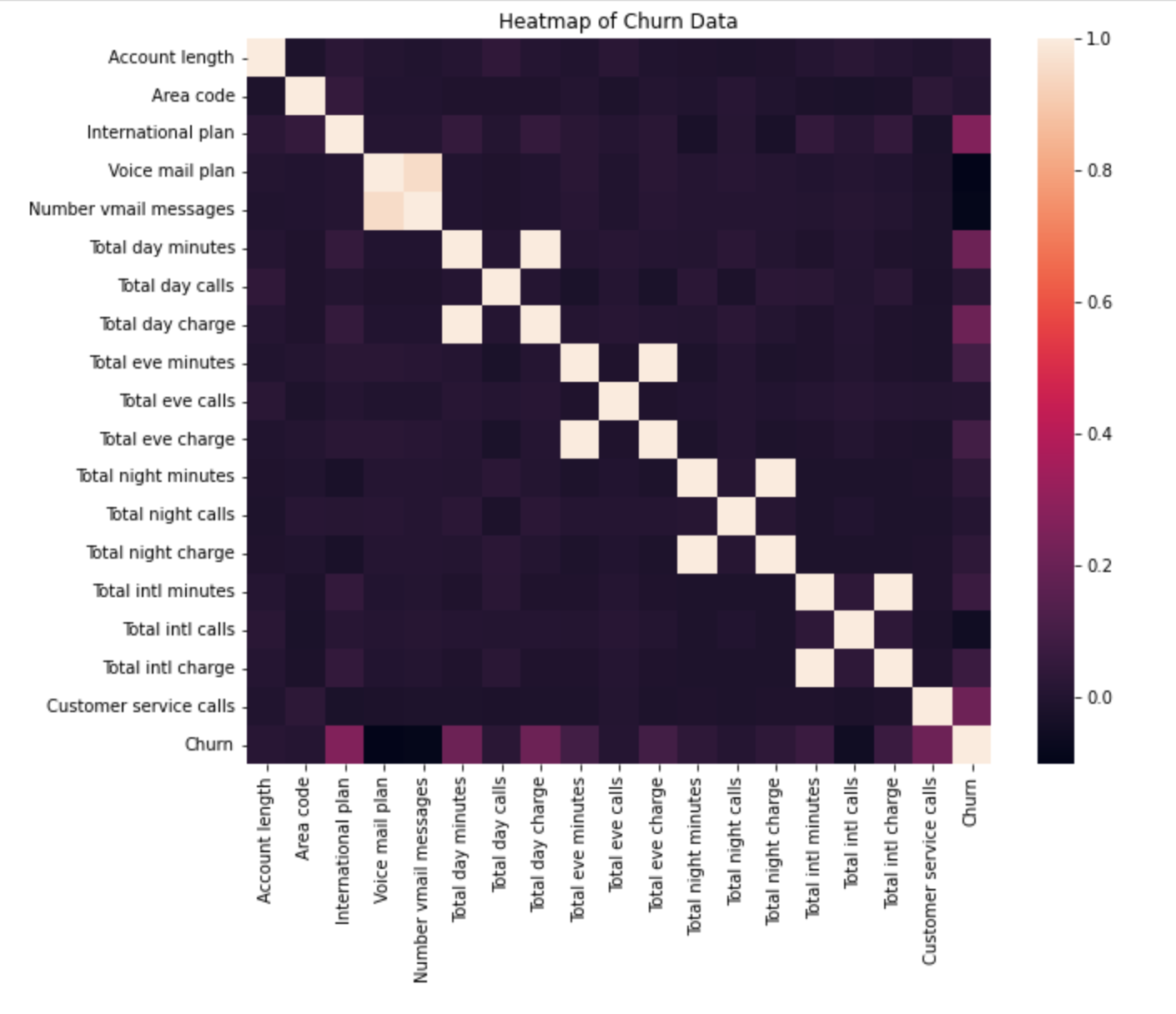
**Conclusion**

Decision Tree performed better when dataset is imbalanced and SVM performed better when it is balanced, both datasets are fairly small! KNN has the best training time in both.

Lois:

**Part I**

The "Churn" column is unbalanced, with 2850 False vs 483 True observations, and the dimensions of the df (3333 x 20) makes Logistic Regression unideal for this data because it won't be able to utilize all the features or handle unbalanced classes well. In considering mean fit time, KNN and SVC takes longer to fit than Logistic Regression, but the row count is not high enough to make this a big concern. Every column in this dataset other than "State" is either already numerical or can easily be converted as such. The heatmap drew on all the data except State shows some multicollinearity among columns, such as Voice mail plan with Number vmail messages and Total day minutes with Total day charge.

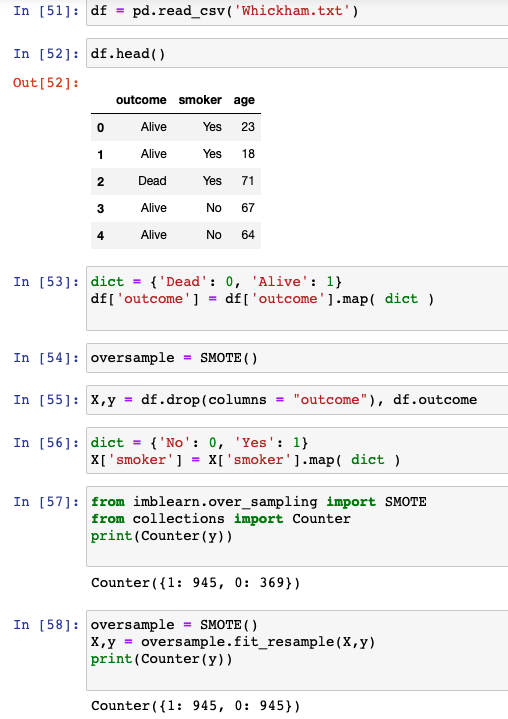


Decision trees handles multicollinearity well and is easy to interpret, but prone to overfitting and more suitable for categorical data. There is also the concern of there being overlapping datapoints, which is very possible due to the number of columms with binary values, and I'm not sure how well the rectangular decision boundaries will fit this data. SVC is not strictly necessary as the number does not exceed the number of training observations, but it can still be a good model, especially since it is capable of drawing nonlinear decision boundaries with the kernel trick. I think SVC and KNN will be the top two models for the churn dataset.

**Part II**

The dimensions of the digits data is 1797 x 64. Logistic Regression takes less time to fit, but SVC and KNN have the highest accuracy for handwritten data classification. SVC is more suited for numerical data, which makes it more appropriate than Decision Trees in this case. Additionally, SVC takes less time to compute, which is why it would be my choice of model for the digits data.

Added a sample SMOTE implementation, this was also covered in Savio live session:



**Install imblearn**

imbalanced-learn is currently available on the PyPi’s repositories and you can install it via pip:

pip install -U --trusted-host pypi.org --trusted-host files.pythonhosted.org imbalanced-learn

pip install -U --trusted-host pypi.org --trusted-host files.pythonhosted.org pip

 ————— o —————