Nanodegree Engenheiro de Machine Learning

Aprendizagem Supervisionada

Project 2: Construindo um Sistema de Intervenção para Estudantes

Bem-vindo ao segundo projeto do Nanodegree de Machine Learning! Neste Notebook, alguns templates de código já foram fornecidos, e será o seu trabalho implementar funcionalidades necessárias para completar este projeto com êxito. Seções que começam com **'Implementação'** no cabeçalho indicam que o bloco de código que se segue precisará de funcionalidades adicionais que você deve fornecer. Instruções serão providenciadas para cada seção e as especificações para cada implementação estarão marcadas no bloco de código com o comando `'TODO'`. Tenha certeza de ler atentamente todas as instruções!

Além do código implementado, haverá questões relacionadas ao projeto e à implementação que você deve responder. Cada seção em que você tem que responder uma questão será antecedida de um cabeçalho **'Questão X'**. Leia atentamente cada questão e escreva respostas completas nas caixas de texto subsequentes que começam com **'Resposta: '**. O projeto enviado será avaliado baseado nas respostas para cada questão e a implementação que você forneceu.

> **Nota:** Células de código e Markdown podem ser executadas utilizando o atalho de teclado **Shift + Enter**. Além disso, as células Markdown podem ser editadas, um clique duplo na célula entra no modo de edição.

Questão 1 - Classificação versus Regressão

Seu objetivo neste projeto é identificar estudantes que possam precisar de intervenção antecipada antes de serem reprovados. Que tipo de problema de aprendizagem supervisionada é esse: classificação ou regressão? Por quê?

Resposta: Este problema de identificar estudantes que possam precisar de intervenção antecipada antes de serem reprovados é um problema de Classificação, pois o tipo de variável alvo apresenta valor discreto e não contínuo.

Observando os Dados

Execute a célula de código abaixo para carregar as bibliotecas de Python necessárias e os dados sobre os estudantes. Note que a última coluna desse conjunto de dados, `'passed'`, será nosso rótulo alvo (se o aluno foi ou não aprovado). As outras colunas são atributos sobre cada aluno.

In [4]:

```
# Importar bibliotecas
import numpy as np
import pandas as pd
from time import time
from sklearn.metrics import f1_score

# Ler os dados dos estudantes
student_data = pd.read_csv("student-data.csv")
print ("Os dados dos estudantes foram lidos com êxito!")
```

```
Os dados dos estudantes foram lidos com êxito!
```

In [5]:

```
student_data
```

Out[5]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...
1	GP	F	17	U	GT3	T	1	1	at_home	other	...
2	GP	F	15	U	LE3	T	1	1	at_home	other	...
3	GP	F	15	U	GT3	T	4	2	health	services	...
4	GP	F	16	U	GT3	T	3	3	other	other	...
5	GP	M	16	U	LE3	T	4	3	services	other	...
6	GP	M	16	U	LE3	T	2	2	other	other	...
7	GP	F	17	U	GT3	A	4	4	other	teacher	...
8	GP	M	15	U	LE3	A	3	2	services	other	...
9	GP	M	15	U	GT3	T	3	4	other	other	...
10	GP	F	15	U	GT3	T	4	4	teacher	health	...
11	GP	F	15	U	GT3	T	2	1	services	other	...
12	GP	M	15	U	LE3	T	4	4	health	services	...
13	GP	M	15	U	GT3	T	4	3	teacher	other	...
14	GP	M	15	U	GT3	A	2	2	other	other	...
15	GP	F	16	U	GT3	T	4	4	health	other	...
16	GP	F	16	U	GT3	T	4	4	services	services	...
17	GP	F	16	U	GT3	T	3	3	other	other	...
18	GP	M	17	U	GT3	T	3	2	services	services	...
19	GP	M	16	U	LE3	T	4	3	health	other	...
20	GP	M	15	U	GT3	T	4	3	teacher	other	...
21	GP	M	15	U	GT3	T	4	4	health	health	...
22	GP	M	16	U	LE3	T	4	2	teacher	other	...
23	GP	M	16	U	LE3	T	2	2	other	other	...
24	GP	F	15	R	GT3	T	2	4	services	health	...
25	GP	F	16	U	GT3	T	2	2	services	services	...
26	GP	M	15	U	GT3	T	2	2	other	other	...
27	GP	M	15	U	GT3	T	4	2	health	services	...
28	GP	M	16	U	LE3	A	3	4	services	other	...
29	GP	M	16	U	GT3	T	4	4	teacher	teacher	...
...	...	...	...	...	...	...	...	...	...	...	...
365	MS	M	18	R	GT3	T	1	3	at_home	other	...
366	MS	M	18	U	LE3	T	4	4	teacher	services	...
367	MS	F	17	R	GT3	T	1	1	other	services	...

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...
368	MS	F	18	U	GT3	T	2	3	at_home	services	...
369	MS	F	18	R	GT3	T	4	4	other	teacher	...
370	MS	F	19	U	LE3	T	3	2	services	services	...
371	MS	M	18	R	LE3	T	1	2	at_home	services	...
372	MS	F	17	U	GT3	T	2	2	other	at_home	...
373	MS	F	17	R	GT3	T	1	2	other	other	...
374	MS	F	18	R	LE3	T	4	4	other	other	...
375	MS	F	18	R	GT3	T	1	1	other	other	...
376	MS	F	20	U	GT3	T	4	2	health	other	...
377	MS	F	18	R	LE3	T	4	4	teacher	services	...
378	MS	F	18	U	GT3	T	3	3	other	other	...
379	MS	F	17	R	GT3	T	3	1	at_home	other	...
380	MS	M	18	U	GT3	T	4	4	teacher	teacher	...
381	MS	M	18	R	GT3	T	2	1	other	other	...
382	MS	M	17	U	GT3	T	2	3	other	services	...
383	MS	M	19	R	GT3	T	1	1	other	services	...
384	MS	M	18	R	GT3	T	4	2	other	other	...
385	MS	F	18	R	GT3	T	2	2	at_home	other	...
386	MS	F	18	R	GT3	T	4	4	teacher	at_home	...
387	MS	F	19	R	GT3	T	2	3	services	other	...
388	MS	F	18	U	LE3	T	3	1	teacher	services	...
389	MS	F	18	U	GT3	T	1	1	other	other	...
390	MS	M	20	U	LE3	A	2	2	services	services	...
391	MS	M	17	U	LE3	T	3	1	services	services	...
392	MS	M	21	R	GT3	T	1	1	other	other	...
393	MS	M	18	R	LE3	T	3	2	services	other	...
394	MS	M	19	U	LE3	T	1	1	other	at_home	...

395 rows × 31 columns

In [6]:

```
student_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 31 columns):
school        395 non-null object
sex           395 non-null object
age           395 non-null int64
address       395 non-null object
famsize       395 non-null object
Pstatus       395 non-null object
Medu          395 non-null int64
Fedu          395 non-null int64
Mjob          395 non-null object
Fjob          395 non-null object
reason        395 non-null object
guardian      395 non-null object
traveltime    395 non-null int64
studytime     395 non-null int64
failures      395 non-null int64
schoolsup     395 non-null object
famsup        395 non-null object
paid          395 non-null object
activities    395 non-null object
nursery       395 non-null object
higher        395 non-null object
internet      395 non-null object
romantic      395 non-null object
famrel        395 non-null int64
freetime      395 non-null int64
goout         395 non-null int64
Dalc          395 non-null int64
Walc          395 non-null int64
health        395 non-null int64
absences      395 non-null int64
passed        395 non-null object
dtypes: int64(13), object(18)
memory usage: 95.7+ KB
```

In [7]:

```
student_data.describe()
```

Out[7]:

	age	Medu	Fedu	traveltime	studytime	failures	famr
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.0
mean	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	3.944
std	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	0.896
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000
25%	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	4.000
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	4.000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000

Implementação: Observando os Dados

Vamos começar observando o conjunto de dados para determinar quantos são os estudantes sobre os quais temos informações e entender a taxa de graduação entre esses estudantes. Na célula de código abaixo, você vai precisar calcular o seguinte:

- O número total de estudantes, `n_students`.
- O número total de atributos para cada estudante, `n_features`.
- O número de estudantes aprovados, `n_passed`.
- O número de estudantes reprovados, `n_failed`.
- A taxa de graduação da classe, `grad_rate`, em porcentagem (%).

In [8]:

```
# TODO: Calcule o número de estudante
n_students = len(student_data)

# TODO: Calcule o número de atributos
n_features=len(student_data.columns[:-1]) #menos a coluna 'passed'

# TODO: Calcule o número de alunos aprovados
#Alunos aprovados > passed = 'yes'
#contador=(student_data['passed'] == 'yes')
#n_passed=contador.sum()
#forma melhor:
n_passed=sum(student_data['passed'] == 'yes')

# TODO: Calcule o número de alunos reprovados
contador1=(student_data['passed'] == 'no')
n_failed=contador1.sum()

# TODO: Calcule a taxa de graduação
grad_rate = (n_passed * 100) / n_students


# Imprima os resultados
print ("Número total de estudantes: {}".format(n_students))
print ("Número de atributos: {}".format(n_features))
print ("Número de estudantes aprovados: {}".format(n_passed))
print ("Número de estudantes reprovados: {}".format(n_failed))
print ("Taxa de graduação: {:.2f}%".format(grad_rate))
```

```
Número total de estudantes: 395
Número de atributos: 30
Número de estudantes aprovados: 265
Número de estudantes reprovados: 130
Taxa de graduação: 67.09%
```

Preparando os Dados

Nesta seção, vamos preparara os dados para modelagem, treinamento e teste.

Identificar atributos e variáveis-alvo

É comum que os dados que você obteve contenham atributos não numéricos. Isso pode ser um problema, dado que a maioria dos algoritmos de machine learning esperam dados númericos para operar cálculos.

Execute a célula de código abaixo para separar os dados dos estudantes em atributos e variáveis-alvo e verificar se algum desses atributos é não numérico.

In [9]:

```
# Extraia as colunas dos atributos
feature_cols = list(student_data.columns[:-1])

# Extraia a coluna-alvo, 'passed'
target_col = student_data.columns[-1]

# Mostre a lista de colunas
print ("Colunas de atributos:\n{}".format(feature_cols))
print ("\nColuna-alvo: {}".format(target_col))

# Separe os dados em atributos e variáveis-alvo (X_all e y_all, respectivamente)
X_all = student_data[feature_cols]
y_all = student_data[target_col]

# Mostre os atributos imprimindo as cinco primeiras linhas
print ("\nFeature values:")
print (X_all.head())
```

```
Colunas de atributos:
['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu',
'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failure
s', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'int
ernet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'healt
h', 'absences']

Coluna-alvo: passed

Feature values:
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob
\
0     GP   F   18       U     GT3       A     4     4  at_home   teacher

1     GP   F   17       U     GT3       T     1     1  at_home     other

2     GP   F   15       U     LE3       T     1     1  at_home     other

3     GP   F   15       U     GT3       T     4     2   health  services

4     GP   F   16       U     GT3       T     3     3    other     other


    ...    higher internet  romantic  famrel  freetime goout Dalc Walc hea
lth  \
0   ...       yes       no        no       4         3     4    1    1
  3
1   ...       yes      yes        no       5         3     3    1    1
  3
2   ...       yes      yes        no       4         3     2    2    3
  3
3   ...       yes      yes       yes       3         2     2    1    1
  5
4   ...       yes       no        no       4         3     2    1    2
  5

  absences
0        6
1        4
2       10
3        2
4        4

[5 rows x 30 columns]
```

Pré-processar Colunas de Atributo

Como você pode ver, há muitas colunas não numéricas que precisam ser convertidas! Muitas delas são simplesmente `yes`/`no`, por exemplo, a coluna `internet`. É razoável converter essas variáveis em valores (binários) `1`/`0`.

Outras colunas, como `Mjob` e `Fjob`, têm mais do que dois valores e são conhecidas como variáveis categóricas. A maneira recomendada de lidar com esse tipo de coluna é criar uma quantidade de colunas proporcional aos possíveis valores (por exemplo, `Fjob_teacher`, `Fjob_other`, `Fjob_services`, etc), e assinalar `1` para um deles e `0` para todos os outros.

Essas colunas geradas são por vezes chamadas de *variáveis postiças* (em inglês: *dummy variables*), e nós iremos utilizar a função `pandas.get_dummies()` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html?highlight=get_dummies#pandas.get_dummies) para fazer essa conversão. Execute a célula de código abaixo para executar a rotina de pré-processamento discutida nesta seção.

In [10]:

```
def preprocess_features(X):
    ''' Pré-processa os dados dos estudantes e converte as variáveis binárias não numéricas em
        variáveis binárias (0/1). Converte variáveis categóricas em variáveis postiças.
    '''

    # Inicialize nova saída DataFrame
    output = pd.DataFrame(index = X.index)

    # Observe os dados em cada coluna de atributos
    for col, col_data in X.iteritems():

        # Se o tipo de dado for não numérico, substitua todos os valores yes/no por 1/0
        if col_data.dtype == object:
            col_data = col_data.replace(['yes', 'no'], [1, 0])

        # Se o tipo de dado for categórico, converta-o para uma variável dummy
        if col_data.dtype == object:
            # Example: 'school' => 'school_GP' and 'school_MS'
            col_data = pd.get_dummies(col_data, prefix = col)

        # Reúna as colunas revisadas
        output = output.join(col_data)

    return output

X_all = preprocess_features(X_all)
print ("Processed feature columns ({} total features):\n{}".format(len(X_all.columns), list(X_all.columns)))
```

```
Processed feature columns (48 total features):
['school_GP', 'school_MS', 'sex_F', 'sex_M', 'age', 'address_R', 'address_
U', 'famsize_GT3', 'famsize_LE3', 'Pstatus_A', 'Pstatus_T', 'Medu', 'Fed
u', 'Mjob_at_home', 'Mjob_health', 'Mjob_other', 'Mjob_services', 'Mjob_te
acher', 'Fjob_at_home', 'Fjob_health', 'Fjob_other', 'Fjob_services', 'Fjo
b_teacher', 'reason_course', 'reason_home', 'reason_other', 'reason_reputa
tion', 'guardian_father', 'guardian_mother', 'guardian_other', 'traveltim
e', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities',
'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goou
t', 'Dalc', 'Walc', 'health', 'absences']
```

Implementação: Divisão dos Dados de Treinamento e Teste

Até agora, nós convertemos todos os atributos *categóricos* em valores numéricos. Para o próximo passo, vamos dividir os dados (tanto atributos como os rótulos correspondentes) em conjuntos de treinamento e teste. Na célula de código abaixo, você irá precisar implementar o seguinte:

- Embaralhe aleatoriamente os dados (`X_all`, `y_all`) em subconjuntos de treinamento e teste.
  - Utilizar 300 pontos de treinamento (aproxidamente 75%) e 95 pontos de teste (aproximadamente 25%).
  - Estabelecer um `random_state` para as funções que você utiliza, se a opção existir.
  - Armazene os resultados em `X_train`, `X_test`, `y_train` e `y_test`.

In [16]:

```
# TODO: Importe qualquer funcionalidade adicional de que você possa precisar aqui
from sklearn.model_selection import train_test_split

# TODO: Estabeleça o número de pontos de treinamento
num_train  = 300

# Estabeleça o número de pontos de teste
num_test = X_all.shape[0] - num_train

# TODO: Emabaralhe e distribua o conjunto de dados de acordo com o número de pontos de
 treinamento e teste abaixo

X_train, X_test, y_train, y_test = train_test_split(X_all, y_all, test_size=num_test, r
andom_state=0,stratify=y_all)
# Mostre o resultado da distribuição
print ("O conjunto de treinamento tem {} amostras.".format(X_train.shape[0]))
print ("O conjunto de teste tem {} amostras.".format(X_test.shape[0]))
```

```
O conjunto de treinamento tem 300 amostras.
O conjunto de teste tem 95 amostras.
```

Treinando e Avaliando Modelos

Nesta seção, você irá escolher 3 modelos de aprendizagem supervisionada que sejam apropriados para esse problema e que estejam disponíveis no `scikit-learn`. Primeiro você irá discutir o raciocínio por trás da escolha desses três modelos considerando suas vantagens e desvantagens e o que você sabe sobre os dados. Depois você irá ajustar o modelo a diferentes tamanhos de conjuntos de treinamento (com 100, 200 e 300 pontos) e medir a pontuação F_1. Você vai precisar preencher três tabelas (uma para cada modelo) que mostrem o tamanho do conjunto de treinamento, o tempo de treinamento, o tempo de previsão e a pontuação F_1 no conjunto de treinamento.

Os seguintes modelos de aprendizagem supervisionada estão atualmente disponíveis no `scikit-learn` (http://scikit-learn.org/stable/supervised_learning.html) para você escolher:

- Gaussian Naive Bayes (GaussianNB)
- Árvores de Decisão
- Métodos de agregação (Bagging, AdaBoost, Random Forest, Gradient Boosting)
- K-Nearest Neighbors (KNeighbors)
- Método do gradiente estocástico (SGDC)
- Máquinas de vetores de suporte (SVM)
- Regressão logística

Questão 2 - Aplicação dos Modelos

Liste três modelos de aprendizagem supervisionada que são apropriadas para esse problema. Para cada modelo escolhido:

- Descreva uma aplicação em mundo real na indústria em que o modelo pode ser aplicado. *(Talvez você precise fazer um pouco de pesquisa para responder essa questão – dê as devidas referências!)*
- Quais são as vantagens do modelo; quando ele tem desempenho melhor?
- Quais são as desvantagens do modelo, quando ele tem desempenho pior?
- O que faz desse modelo um bom candidato para o problema, considerando o que você sabe sobre os dados?

Resposta:

1.Árvores de decisão a)Este modelo é aplicado na área de educação, a partir de um conjunto de dados retirados de questionários de provas, é possível classificar qual é a área da questão, ex: biologia, história, e também qual é o tema que está sendo abordado nesta questão. b)As vantages são: Árvores de decisão são simples de entender e interpretar, podem ser visualizadas e requerem pouca preparação de dados. Tem desempenho melhor quando configura-se o mínimo de amostras requeridas no nó do node ou configura-se o max_depth da árvore para evitar o problema de overfitting. c)A partir do momento que a árvore vai cada vez mais crescendo ela tende a ter problema de sobreajuste. d)Até o momento, sei que será necessário analisar se um estudante precisará de intervenção ou não com base nos dados que se tem. Na minha visão, esse modelo seria um bom candidato, pois por sua abordagem top-down em forma de árvore de decisão, escolhe-se o melhor atributo que dividirá o conjunto de dados, e assim vai se estreitando, até chegar na resposta que se deseja, que neste é se o aluno precisa ou não de intervenção.

2.SVM a) Uma aplicação no mundo real seria a classificação dos gostos de filmes, por exemplo, com base no perfil do cliente. b) As vantagens do modelo são: Svms trabalham muito bem em dominios complicados em que existe uma clara margem de separação, mas não funcionam tão bem em conjunto de dados muito grandes.É possível também alterar o kernel, visando melhor desempenho do modelo.Ele tem desempenho melhor quando o conjunto de dados é pequeno. c)Ele tem desenpenho pior quando o conjunto de dados é muito grande e tem muitos ruídos nos dados. d)Esse modelo serve para classificação e é um bom candidato, poís tem bom desempenho quando o conjunto de dados é pequeno.

3.Nayve Bayes a)Uma aplicação no mundo real seria a construção de modelos para análise de sentimentos, filtragem de spam e classificação de textos. b)As vantagens do modelo são: O modelo Naive Bayes é fácil de construir e particularmente útil para grandes conjuntos de dados.Tem desempenho melhor quando: "é usado principalmente em classificação de texto e com os problemas que têm múltiplas classes." E "é fácil e rápido para prever o conjunto de dados da classe de teste. Também tem um bom desempenho na previsão de classes múltiplas." c)As desvantagens do modelo são: -"Se a variável categórica tem uma categoria (no conjunto de dados de teste) que não foi observada no conjunto de dados de treinamento, então o modelo irá atribuir uma probabilidade de 0 (zero) e não será capaz de fazer uma previsão. Isso é muitas vezes conhecido como “Zero Frequency”." -"Outra limitação do Naive Bayes é a suposição de preditores independentes. Na vida real, é quase impossível que ter um conjunto de indicadores que sejam completamente independentes." d)Esse modelo serve para classificação e pretendo usa-lo para comparar seu desempenho com o svm, pois diferente do svm, este algoritmo tem um desempenho melhor para um conjunto de dados grande.

- Referências:
- O próprio curso Nanodegree Machine Learning
- Documentação Sklearn
- Palestra da Startup AppProva(Aplicação de árvore de decisão)
- https://www.vooo.pro/insights/6-passos-faceis-para-aprender-o-algoritmo-naive-bayes-com-o-codigo-em-python/ (https://www.vooo.pro/insights/6-passos-faceis-para-aprender-o-algoritmo-naive-bayes-com-o-codigo-em-python/)

Configuração

Execute a célula de código abaixo para inicializar três funções de ajuda que você pode utilizar para treinar e testar os três modelos de aprendizagem supervisionada que você escolheu acima. As funções são as seguintes:

- `train_classifier` - recebe como parâmetro um classificador e dados de treinamento e ajusta o classificador aos dados.
- `predict_labels` - recebe como parâmetro um classificador ajustado, atributos e rótulo alvo e faz estimativas utilizando a pontuação do F_1.
- `train_predict` - recebe como entrada um classificador, e dados de treinamento e teste, e executa `train_clasifier` e `predict_labels`.
    - Essa função vai dar a pontuação F_1 tanto para os dados de treinamento como para os de teste, separadamente.

In [12]:

```
def train_classifier(clf, X_train, y_train):
    ''' Ajusta um classificador para os dados de treinamento. '''

    # Inicia o relógio, treina o classificador e, então, para o relógio
    start = time()
    clf.fit(X_train, y_train)
    end = time()

    # Imprime os resultados
    print ("O modelo foi treinado em {:.4f} segundos".format(end - start))


def predict_labels(clf, features, target):
    ''' Faz uma estimativa utilizando um classificador ajustado baseado na pontuação F
1. '''

    # Inicia o relógio, faz estimativas e, então, o relógio para
    start = time()
    y_pred = clf.predict(features)
    end = time()

    # Imprime os resultados de retorno
    print ("As previsões foram feitas em {:.4f} segundos.".format(end - start))
    return f1_score(target.values, y_pred, pos_label='yes')


def train_predict(clf, X_train, y_train, X_test, y_test):
    ''' Treina e faz estimativas utilizando um classificador baseado na pontuação do F
1. '''

    # Indica o tamanho do classificador e do conjunto de treinamento
    print ("Treinando um {} com {} pontos de treinamento. . .".format(clf.__class__.__n
ame__, len(X_train)))

    # Treina o classificador
    train_classifier(clf, X_train, y_train)

    # Imprime os resultados das estimativas de ambos treinamento e teste
    print ("Pontuação F1 para o conjunto de treino: {:.4f}.".format(predict_labels(clf,
 X_train, y_train)))
    print ("Pontuação F1 para o conjunto de teste: {:.4f}.".format(predict_labels(clf,
X_test, y_test)))
```

Implementação: Métricas de Desempenho do Modelo

Com as funções acima, você vai importar os três modelos de aprendizagem supervisionada de sua escolha e executar a função `train_prediction` para cada um deles. Lembre-se de que você vai precisar treinar e usar cada classificador para três diferentes tamanhos de conjuntos de treinamentos: 100, 200 e 300 pontos. Então você deve ter 9 saídas diferentes abaixo – 3 para cada modelo utilizando cada tamanho de conjunto de treinamento. Na célula de código a seguir, você deve implementar o seguinte:

- Importe os três modelos de aprendizagem supervisionada que você escolheu na seção anterior.
- Inicialize os três modelos e armazene eles em `clf_A`, `clf_B` e `clf_C`.
  - Defina um `random_state` para cada modelo, se a opção existir.
  - **Nota:** Utilize as configurações padrão para cada modelo – você vai calibrar um modelo específico em uma seção posterior.
- Crie diferentes tamanhos de conjuntos de treinamento para treinar cada modelo.
  - *Não embaralhe e distribua novamente os dados! Os novos pontos de treinamento devem ser tirados de `X_train` e `y_train`.*
- Treine cada modelo com cada tamanho de conjunto de treinamento e faça estimativas com o conjunto de teste (9 vezes no total).
  Nota: Três tabelas são fornecidas depois da célula de código a seguir, nas quais você deve anotar seus resultados.

In [29]:

```
# TODO: Importe os três modelos de aprendizagem supervisionada do sklearn
# from sklearn import model_A
from sklearn import svm
# from sklearn import model_B
from sklearn.naive_bayes import GaussianNB
# from skearln import model_C
from sklearn import tree

# TODO: Inicialize os três modelos

clf_A = svm.SVC(random_state=0)
train_classifier(clf_A,X_train, y_train)

clf_B = gnb = GaussianNB()
train_classifier(clf_B,X_train, y_train)

clf_C=tree.DecisionTreeClassifier(random_state=0)
train_classifier(clf_C,X_train, y_train)


# TODO: Configure os tamanho dos conjuntos de treinamento
#X_train_100 = 100
#y_train_100 = X_all.shape[0] - X_train_100
#X_train, X_test, y_train, y_test = train_test_split(X_all, y_all, test_size=y_train_10
0, random_state=0)

#X_train_200 = 200
#y_train_200 = X_all.shape[0] - X_train_200
#X_train, X_test, y_train, y_test = train_test_split(X_all, y_all, test_size=y_train_20
0, random_state=0)

#X_train_300 = 300
#y_train_300 = X_all.shape[0] - X_train_300
#X_train, X_test, y_train, y_test = train_test_split(X_all, y_all, test_size=y_train_30
0, random_state=0)

# TODO: Executar a função 'train_predict' para cada classificador e cada tamanho de con
junto de treinamento
##train_predict(clf, X_train, y_train, X_test, y_test)

#y_pred2=predict_labels(clf_A, X_train, y_train)
#predict_labels(clf_B, X_all, y_all)
#predict_labels(clf_C, X_all, y_all)

#train_predict(clf_A, X_train, y_train, X_test, y_test)
#train_predict(clf_B, X_train, y_train, X_test, y_test)
#train_predict(clf_C, X_train, y_train, X_test, y_test)

#Código fornecido pelo revisor "@wittmannf no slack" da Udacity
for clf in [clf_A,clf_B,clf_C]:
    for size in [100, 200, 300]:
        train_predict(clf, X_train[:size], y_train[:size], X_test, y_test)
    print (*[80]) # imprime linha
```

```
O modelo foi treinado em 0.0000 segundos
O modelo foi treinado em 0.0000 segundos
O modelo foi treinado em 0.0000 segundos
Treinando um SVC com 100 pontos de treinamento. . .
O modelo foi treinado em 0.0000 segundos
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de treino: 0.8919.
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de teste: 0.7737.
Treinando um SVC com 200 pontos de treinamento. . .
O modelo foi treinado em 0.0156 segundos
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de treino: 0.8795.
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de teste: 0.8182.
Treinando um SVC com 300 pontos de treinamento. . .
O modelo foi treinado em 0.0156 segundos
As previsões foram feitas em 0.0156 segundos.
Pontuação F1 para o conjunto de treino: 0.8664.
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de teste: 0.8289.
80
Treinando um GaussianNB com 100 pontos de treinamento. . .
O modelo foi treinado em 0.0000 segundos
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de treino: 0.5833.
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de teste: 0.5161.
Treinando um GaussianNB com 200 pontos de treinamento. . .
O modelo foi treinado em 0.0156 segundos
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de treino: 0.8118.
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de teste: 0.7576.
Treinando um GaussianNB com 300 pontos de treinamento. . .
O modelo foi treinado em 0.0000 segundos
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de treino: 0.8000.
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de teste: 0.7273.
80
Treinando um DecisionTreeClassifier com 100 pontos de treinamento. . .
O modelo foi treinado em 0.0000 segundos
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de treino: 1.0000.
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de teste: 0.7107.
Treinando um DecisionTreeClassifier com 200 pontos de treinamento. . .
O modelo foi treinado em 0.0000 segundos
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de treino: 1.0000.
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de teste: 0.6885.
Treinando um DecisionTreeClassifier com 300 pontos de treinamento. . .
O modelo foi treinado em 0.0000 segundos
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de treino: 1.0000.
As previsões foram feitas em 0.0000 segundos.
Pontuação F1 para o conjunto de teste: 0.6875.
80
```

Resultados Tabulados

Edite a célula abaixo e veja como a tabela pode ser desenhada em Markdown (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#tables). Você deve salvar seus resultados abaixo nas tabelas fornecidas.

Classificador 1 - SVM

Tamanho do Conjunto de Treinamento	Tempo de Treinamento	Tempo de Estimativa (teste)	Pontuação F1 (treinamento)	Pontuação F1 (teste)
100	0.0156	0.0000	0.8690	0.7824
200	0.0000	0.0000	0.8765	0.7781
300	0.0156	0.0000	0.8692	0.7586

Classificador 2 - Nayve Bayes

Tamanho do Conjunto de Treinamento	Tempo de Treinamento	Tempo de Estimativa (teste)	Pontuação F1 (treinamento)	Pontuação F1 (teste)
100	0.0000	0.0469	0.4468	0.3636
200	0.0000	0.0000	0.7926	0.7331
300	0.0156	0.0156	0.8088	0.7500

Classificador 3 - Árvore de decisão

Tamanho do Conjunto de Treinamento	Tempo de Treinamento	Tempo de Estimativa (teste)	Pontuação F1 (treinamento)	Pontuação F1 (teste)
100	0.0156	0.0000	1000	0.6997
200	0.0156	0.0000	1000	0.7422
300	0.0000	0.0000	1000	0.7059

Escolhendo o Melhor Modelo

Nesta seção final, você irá escolher dos três modelos de aprendizagem supervisionada o *melhor* para utilizar os dados dos estudantes. Você então executará um busca em matriz otimizada para o modelo em todo o conjunto de treinamento (`X_train` e `y_train`) ao calibrar pelo menos um parâmetro, melhorando em comparação a pontuação F_1 do modelo não calibrado.

Questão 3 - Escolhendo o Melhor Modelo

Baseando-se nos experimentos que você executou até agora, explique em um ou dois parágrafos ao conselho de supervisores qual modelo que você escolheu como o melhor. Qual modelo é o mais apropriado baseado nos dados disponíveis, recursos limitados, custo e desempenho?

Resposta: O modelo mais apropriado é o SVM, pois em relação aos outros, é o que obteve uma pequena diferença entre as pontuações F1 dos conjuntos de treinamento e teste, e também é o que obteve o maior score F1 quando o modelo foi treinando com 300 dados.Em relação ao desempenho, svm é um algoritmo que tem boa performance com conjuntos de dados pequenos, por isso o custo e tempo de execução foram pequenos. Em relação ao erro de viés, pelo F1, podemos perceber que não há erro de viés,pois não há distância grande entre os valores de score. E também não há erro de variância,pois os valores não se encontram tão próximos.

Questão 4 – O Modelo para um Leigo

Em um ou dois parágrafos, explique para o conselho de supervisores, utilizando termos leigos, como o modelo final escolhido deve trabalhar. Tenha certeza que você esteja descrevendo as melhores qualidades do modelo, por exemplo, como o modelo é treinado e como ele faz uma estimativa. Evite jargões técnicos ou matemáticos, como descrever equações ou discutir a implementação do algoritmo.

Resposta: O modelo final escolhido deve traçar uma linha que vai dividir as duas classes de "passed", ou seja, vai dividir os estudantes que passaram e não passaram com base no modelo construído. Esta linha dará a margem máxima.As qualidades do modelo são: ele apresentam os vetores de suporte e o kernel trick. Os vetores de suporte são os pontos de dados de entrada,ou seja, dados necessários para definir o separador de margem máxima. E o kernel trick permite encontrar a melhor margem, assim alterando-se os kernels é possível encontrar um separador linear que tenha a Maior margem e também encontrar uma linha de seperação(*hiperplano entre dados de suas classes). Por fim, o objetivo do svm é maximizar a robustez do resultado.Assim sendo, ele coloca em 1 lugar a classificação correta dos labels e depois leva em consideração a margem, linha.

Outras observações:O modelo é treinado a partir da função .fit , esta função pega os dados do conjunto de treinamento(X_train,y_train). Em relação a estimativa, o modelo pode fazer uma estimativa utilizando a função .predict que vai utilizar o conjunto de teste(X_test,y_test) para prever e visualizar os valores estimados. Após fazer isto, pode-se avaliar o modelo chamando o método .score e avaliando se a precisão do modelo está boa para o modelo proposto.

Implementação: Calibrando o Modelo

Calibre o modelo escolhido. Utilize busca em matriz (`GridSearchCV`) com, pelo menos, um parâmetro importante calibrado com, pelo menos, 3 valores diferentes. Você vai precisar utilizar todo o conjunto de treinamento para isso. Na célula de código abaixo, você deve implementar o seguinte:

- Importe `sklearn.grid_search.gridSearchCV` (http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html) e `sklearn.metrics.make_scorer` (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.make_scorer.html).
- Crie o dicionário de parâmetros que você deseja calibrar para o modelo escolhido.
  - Examplo: `parameters = {'parameter' : [list of values]}`.
- Inicialize o classificador que você escolheu e armazene-o em `clf`.
- Crie a função de pontuação F_1 utilizando `make_scorer` e armazene-o em `f1_scorer`.
  - Estabeleça o parâmetro `pos_label` para o valor correto!
- Execute uma busca em matriz no classificador `clf` utilizando o `f1_scorer` como método de pontuação e armazene-o em `grid_obj`.
- Treine o objeto de busca em matriz com os dados de treinamento (`X_train`, `y_train`) e armazene-o em `grid_obj`.

In [91]:

```
# TODO: Importe 'GridSearchCV' e 'make_scorer'
#import sklearn.grid_search.gridSearchCV
from sklearn.model_selection import GridSearchCV
#import sklearn.metrics.make_scorer
from sklearn.metrics import make_scorer
from sklearn.svm import SVC
# TODO: Crie a lista de parâmetros que você gostaria de calibrar
#parameters = {'kernel:' ['linear', 'poly','sigmoid']}
parameters = {'kernel':('poly','rbf','sigmoid'),'degree':range(2,3),'C':range(1,2)}

#parameters = {'kernel':('linear', 'rbf'),
#{'kernel':('linear', 'rbf'), 'C':[1, 10]}

# TODO: Inicialize o classificador
clf = svm.SVC(random_state=0)
#clf.fit(X_train, y_train)

# TODO: Faça uma função de pontuação f1 utilizando 'make_scorer'
#y_pred = clf.(X_test)
#y_pred = clf.predict(X_test)
#f1_scorer =make_scorer(f1_score(y_test, y_pred, pos_label='yes'))
f1_scorer = make_scorer(f1_score, pos_label="yes",greater_is_better=False)
#f1_scorer = make_scorer(f1_score(y_true, y_pred,pos_label='yes'))


# TODO: Execute uma busca em matriz no classificador utilizando o f1_scorer como método
 de pontuação
grid_obj = GridSearchCV(estimator=clf, param_grid=parameters,scoring=f1_scorer)

# TODO: Ajuste o objeto de busca em matriz para o treinamento de dados e encontre os pa
râmetros ótimos
grid_obj = grid_obj.fit(X_train, y_train)

# Get the estimator
clf =grid_obj.best_estimator_
#grid_obj.best_estimator_
#grid_obj.best_estimator_
# Reporte a pontuação final F1 para treinamento e teste depois de calibrar os parâmetro
sprint "Tuned model has a training F1 score of {:.4f}.".format(predict_labels(clf, X_tr
ain, y_train))
print ("O modelo calibrado tem F1 de {:.4f} no conjunto de treinamento.".format(predict
_labels(clf, X_train, y_train)))
print ("O modelo calibrado tem F1 de {:.4f} no conjunto de teste.".format(predict_label
s(clf, X_test, y_test)))
```

```
As previsões foram feitas em 0.0000 segundos.
O modelo calibrado tem F1 de 0.8879 no conjunto de treinamento.
As previsões foram feitas em 0.0156 segundos.
O modelo calibrado tem F1 de 0.7770 no conjunto de teste.
```

Questão 5 - Pontuação F_1 Final

Qual é a pontuação F_1 do modelo final para treinamento e teste? Como ele se compara ao modelo que não foi calibrado?

Resposta: Comparando-se com o conjunto de dados de treinamento = 300, a pontuação F1 do modelo final para treinamento e teste é maior do que a do modelo que não foi calibrado.

> **Nota**: Uma vez que você completou todas as implementações de código e respondeu todas as questões acima com êxito, você pode finalizar seu trabalho exportando o iPython Nothebook como um document HTML. Você pode fazer isso utilizando o menu acima e navegando para
> **File -> Download as -> HTML (.html)**. Inclua a documentação final junto com o notebook para o envio do seu projeto.