



**FACULDADE DE INFORMÁTICA E ADMINISTRAÇÃO PAULISTA**  
**MBA EM BIG DATA**

## **Projeto de conclusão da disciplina Big Data Mining & Inteligência Artificial**

Professor: Luiz Carlos Barboza Junior

Aluno: Pedro Mauro Monnerat  
RM: 333096

# 1. Introdução

Neste projeto, objetiva-se a aplicação prática de dois algoritmos de inteligência artificial, com a finalidade de diminuir a rotatividade de funcionários em uma empresa.

## 2. Base de dados

A base de dados selecionada para este projeto é uma base distribuída gratuitamente pela IBM, disponibilizada na plataforma Kaggle, onde pode ser encontrada no seguinte link: <https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset>.

O nome da base de dados é *HR Analytics Employee Attrition & Performance*, e ela contém dados relacionados a recursos humanos de uma empresa fictícia.

Através de sua exploração, propõe-se extrair informações relativas a rotatividade de funcionários na empresa, uma vez que esta base apresenta questões importantes relacionadas aos funcionários, como por exemplo a distância de casa para o trabalho, o número de anos desde a última promoção, a quantidade de treinamentos recebidos no último ano, etc.

Esta base de dados contém 1.470 entradas, que foram divididas de maneira randômica, na proporção de 70% de entradas para base de treino, e 30% de entradas para a base de teste.

### 3. Análise de negócio

Um problema comum a todas as empresas é o de determinar como diminuir a rotatividade de funcionários.

A razão deste problema é que a rotatividade de funcionários, também conhecida como *turnover*, que pode ser definido como um elevado índice de demissão e contratação de funcionários, trazendo prejuízos para as empresas. Isto se dá por uma série de razões, como a redução momentânea de funcionários, os altos custos envolvidos em demissões e contratações e a perda de talentos. Todos estes fatores, em conjunto, acarretam a uma queda de produtividade generalizada, levando a empresa a ter baixa performance.

Portanto, fica claro que é desejável às empresas conseguir reduzir a rotatividade de funcionários, evitando assim evitar gastos e reter talentos, melhorando a performance da empresa como um todo.

Para se compreender os motivos pelos quais os funcionários são levados a abandonar seus empregos, podemos investigar os dados relacionados aos mesmos. Estes dados podem ser encontrados nos sistemas de recursos humanos.

Neste projeto, propomos a aplicação de dois algoritmos de inteligência artificial distintos, para tentar responder os seguintes problemas de negócios.

1. Identificar funcionários mais propícios a deixar a empresa, baseando-se na totalidade de dados presentes na base de recursos humanos. Para resolver este problema, aplicaremos o algoritmo de classificação supervisionado KNN (*K-Nearest Neighbors Algorithm*).
2. Identificar funcionários mais propícios a deixar a empresa baseando-se exclusivamente em poucas variáveis categóricas que podem ser obtidas pela empresa através de uma enquete. Para resolver este problema, aplicaremos o algoritmo de redes neurais.

## 4. Análise exploratória de dados

O primeiro passo executado foi a importação da fonte de dados.

### Importação da fonte de dados

```
import pandas as pd
treino = pd.read_csv('/content/drive/My Drive/PROJ/dados/ibm_treino.csv')
treino
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction
0	42	Yes	Travel_Frequently	933	Research & Development	19	3	Medical	1	752	
1	38	No	Travel_Rarely	330	Research & Development	17	1	Life Sciences	1	1088	
2	43	No	Travel_Frequently	559	Research & Development	10	4	Life Sciences	1	448	
3	31	No	Non-Travel	587	Sales	2	4	Life Sciences	1	1324	
4	28	No	Travel_Frequently	791	Research & Development	1	4	Medical	1	1286	
...	...	...	...	...	...	...	...	...	...	...	...
1024	31	No	Travel_Rarely	1062	Research & Development	24	3	Medical	1	1252	

Em seguida iniciou-se uma análise exploratória dos dados, com a finalidade de se ter um melhor entendimento dos dados disponíveis.

Neste passo, foi inicialmente executado o comando *describe()*, e posteriormente houve a plotagem de *value counts* de cada uma das variáveis disponíveis.

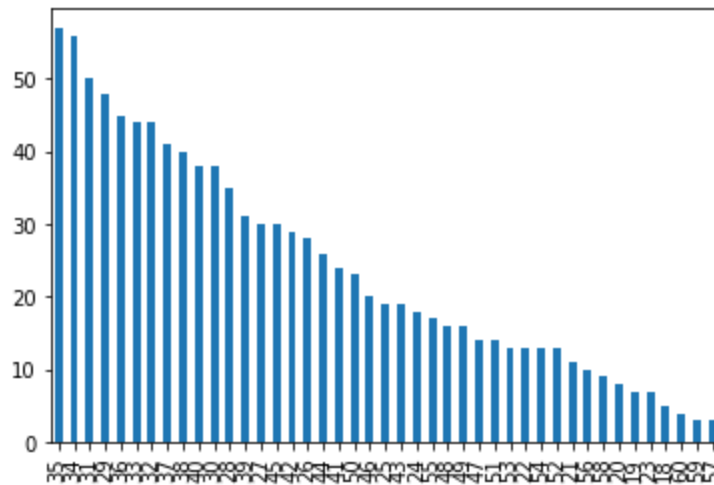
### Análise exploratória dos dados

```
treino.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate
count	1029.000000	1029.000000	1029.000000	1029.000000	1029.0	1029.000000	1029.000000	1029.000000
mean	36.824101	806.881438	9.303207	2.911565	1.0	1009.380952	2.706511	66.203110
std	9.094786	399.603402	8.220182	1.038628	0.0	607.097897	1.101540	20.227365
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000

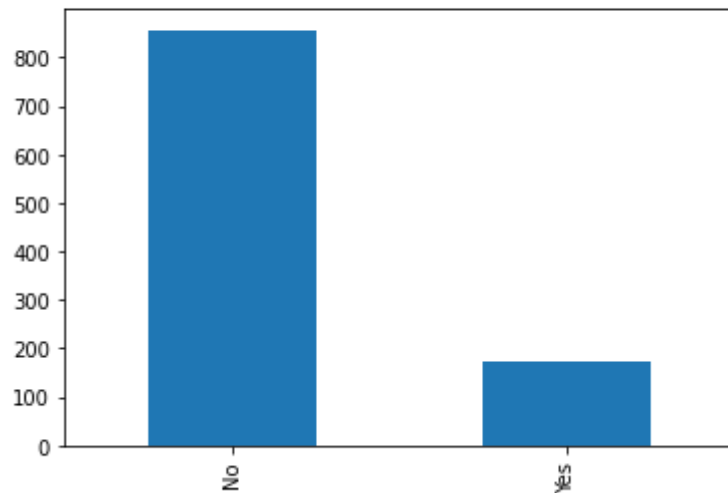
```
▶ treino['Age'].value_counts().plot.bar()
```

```
↳ <matplotlib.axes._subplots.AxesSubplot at 0x7fbbab3c2b0>
```




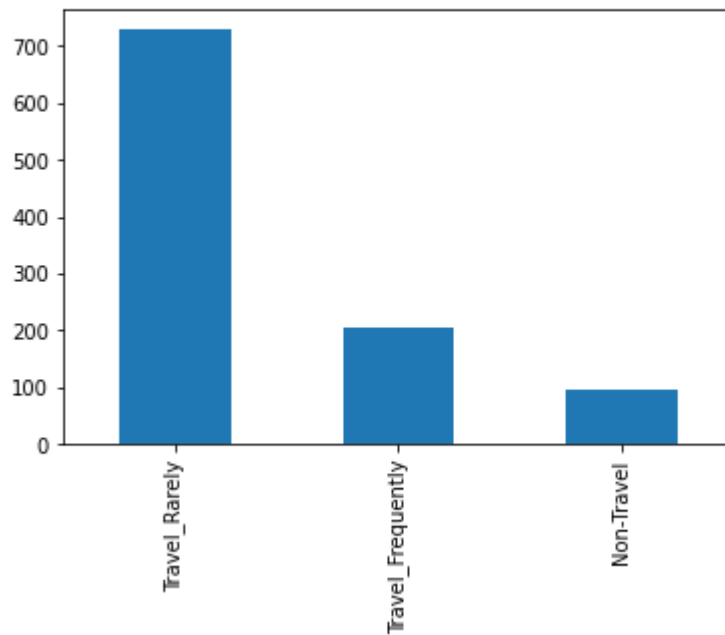
```
▶ treino['Attrition'].value_counts().plot.bar()
```


```
↳ <matplotlib.axes._subplots.AxesSubplot at 0x7fbbab5f80>
```




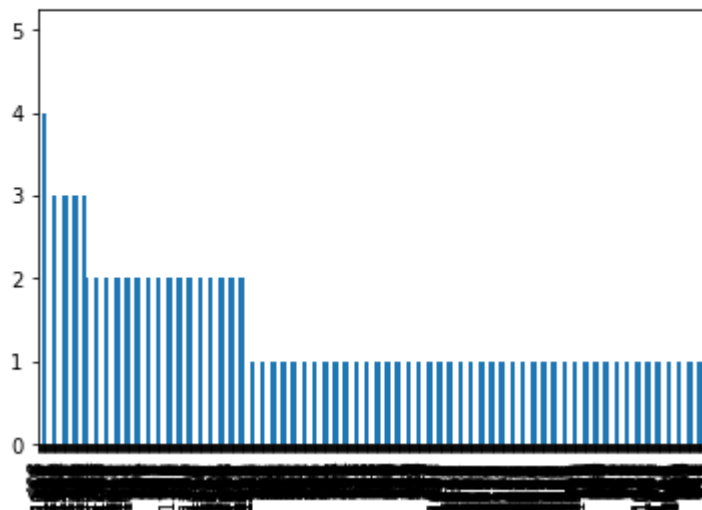
```
[ ] treino['BusinessTravel'].value_counts().plot.bar()
```

 <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbbaa55a320>



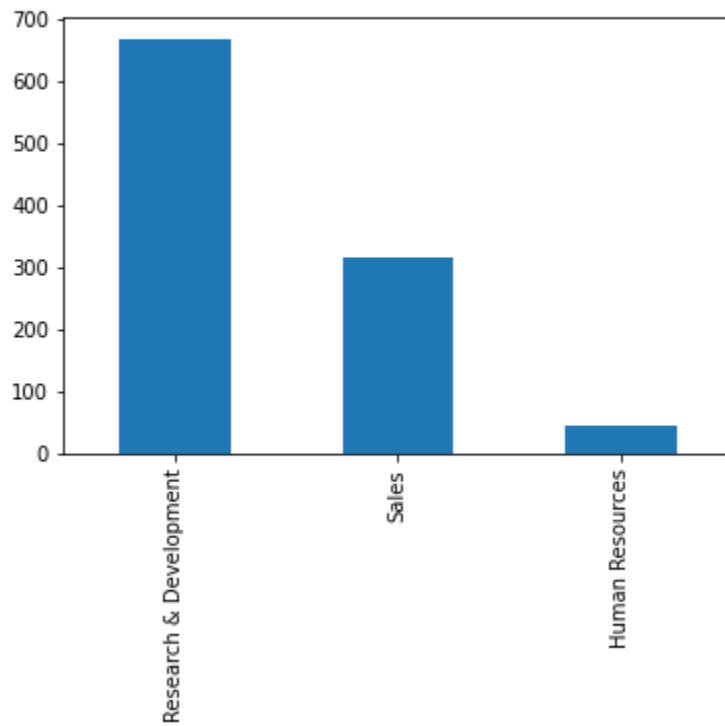
 treino['DailyRate'].value\_counts().plot.bar()

 <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbbaa4d30>



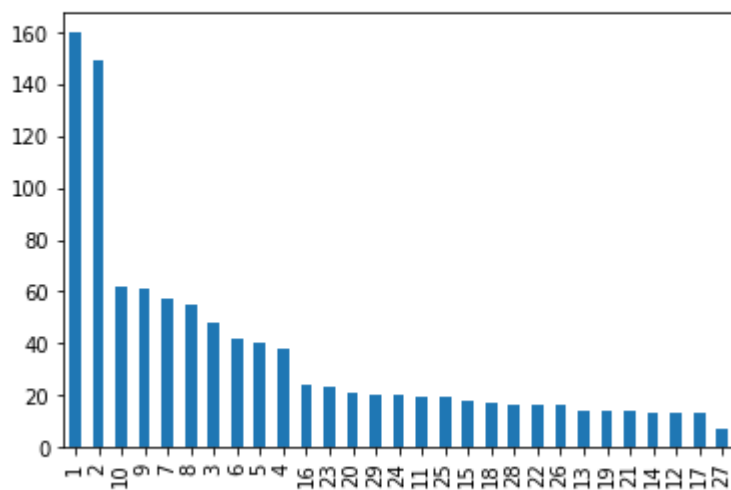
```
[ ] treino['Department'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba9811630>



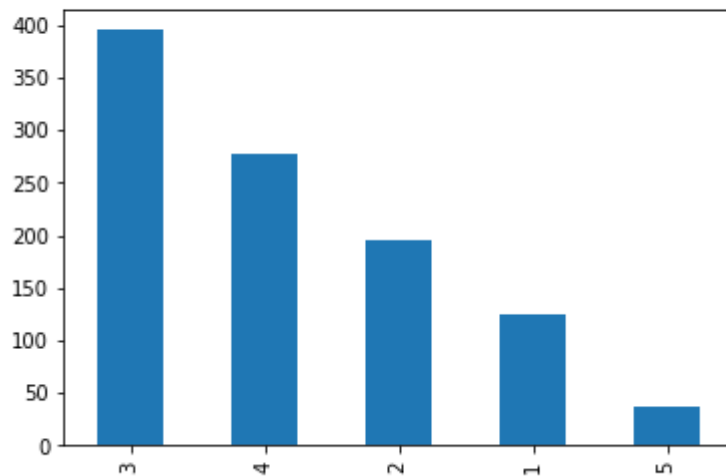
```
[ ] treino['DistanceFromHome'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba97766a0>



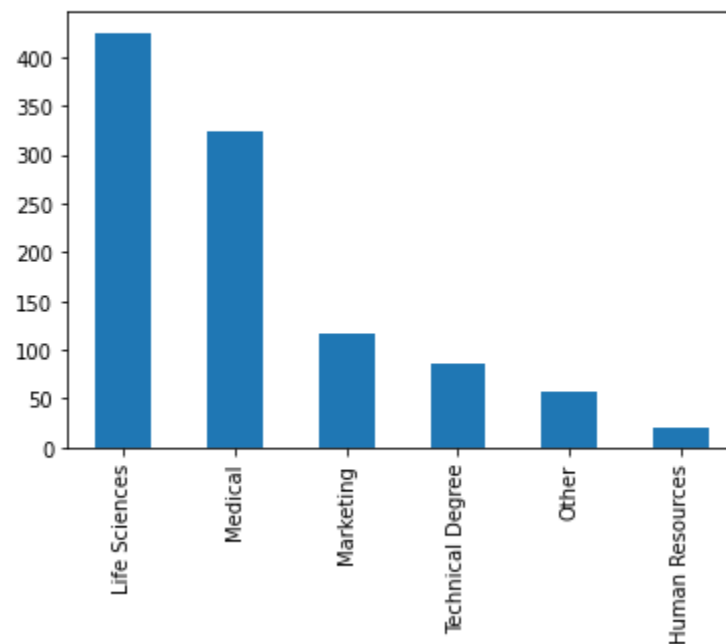
```
[ ] treino['Education'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba97769e8>



```
[ ] treino['EducationField'].value_counts().plot.bar()
```

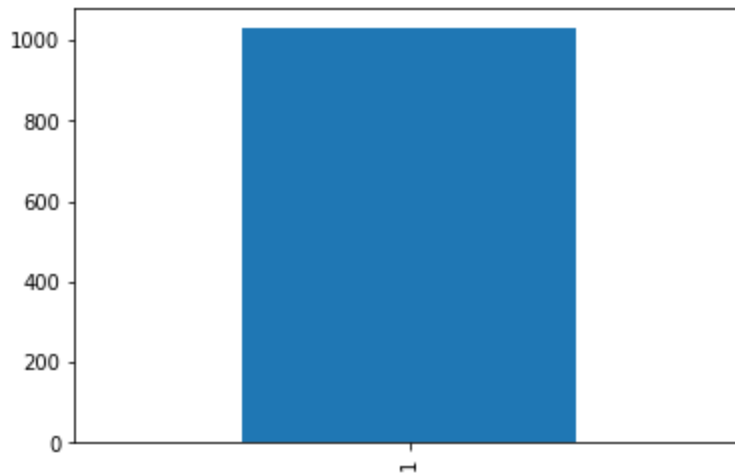
↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba94cd940>





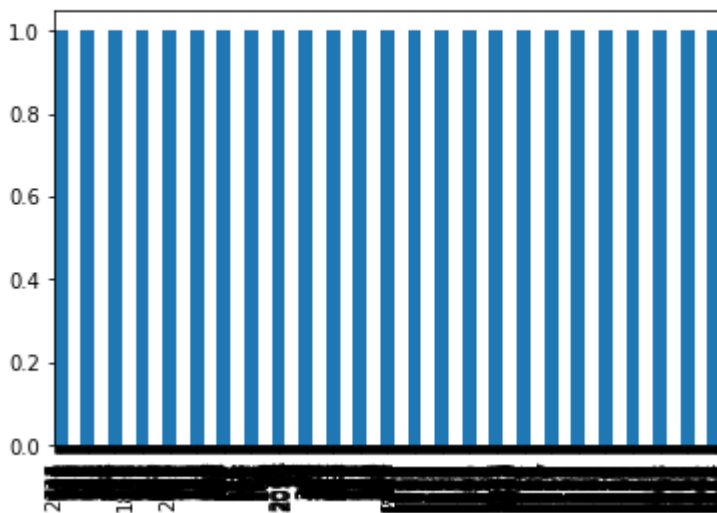
```
[ ] treino['EmployeeCount'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba941cac8>



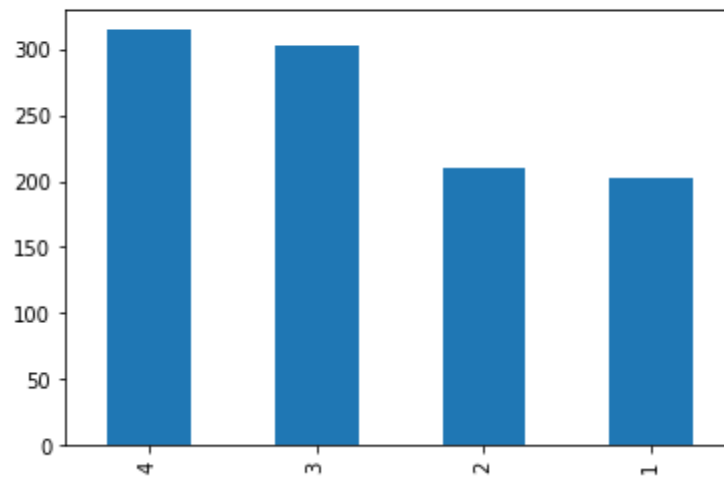
```
[ ] treino['EmployeeNumber'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba978afd0>



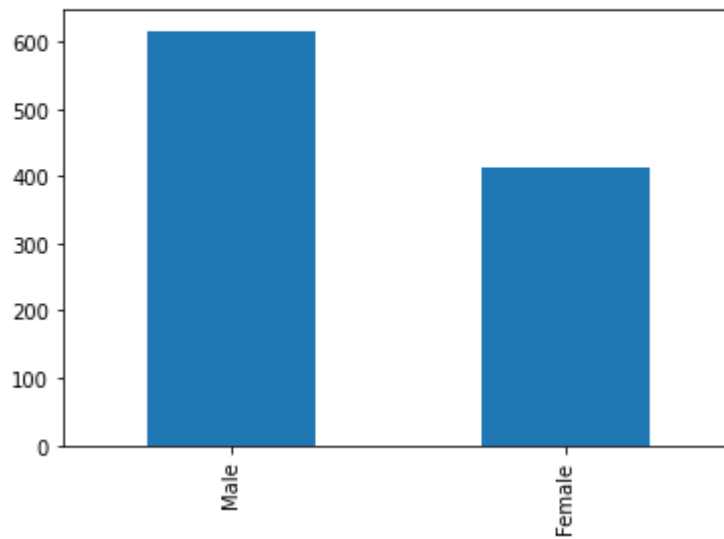
```
[ ] treino['EnvironmentSatisfaction'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbb928fc18>



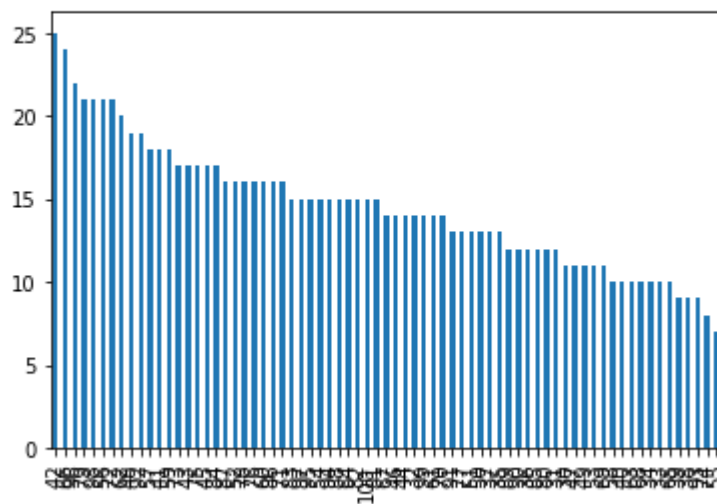
```
[ ] treino['Gender'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbb80e64a8>



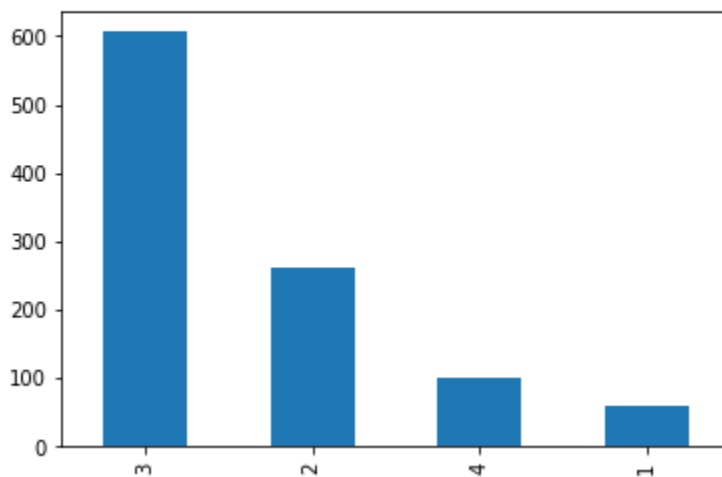
```
[ ] treino['HourlyRate'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbb7cc8320>



```
[ ] treino['JobInvolvement'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbb7d42438>

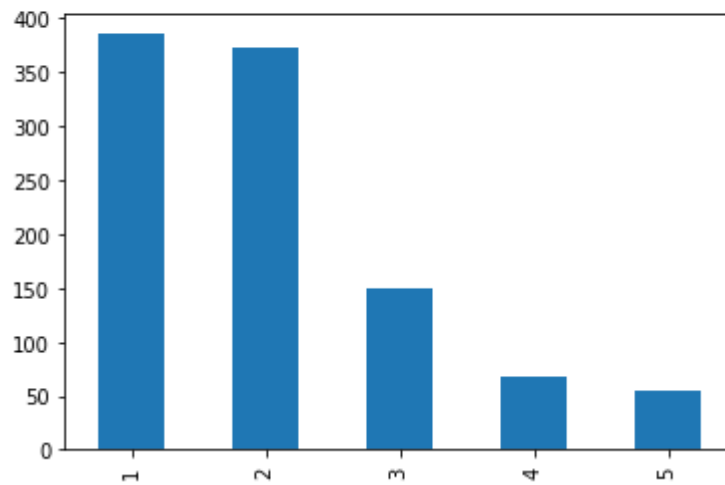




```
treino['JobLevel'].value_counts().plot.bar()
```



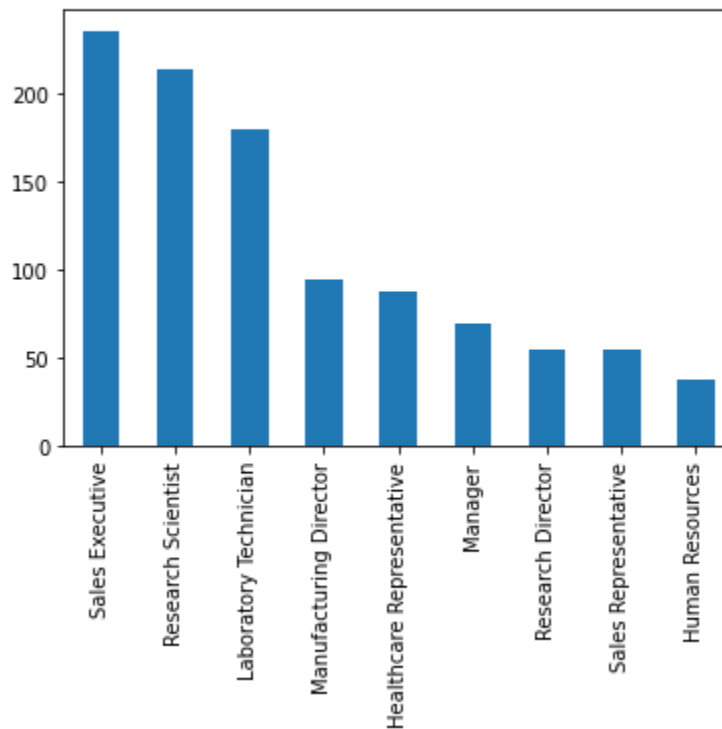
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb7b42c88>
```



```
[ ] treino['JobRole'].value_counts().plot.bar()
```

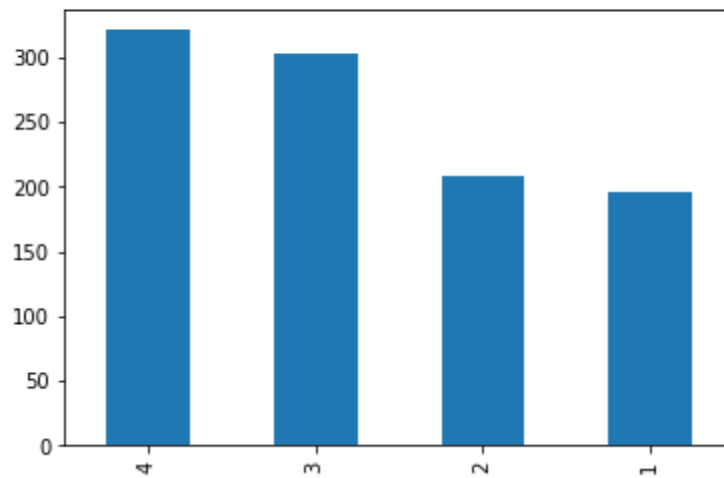


```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb7afcda0>
```



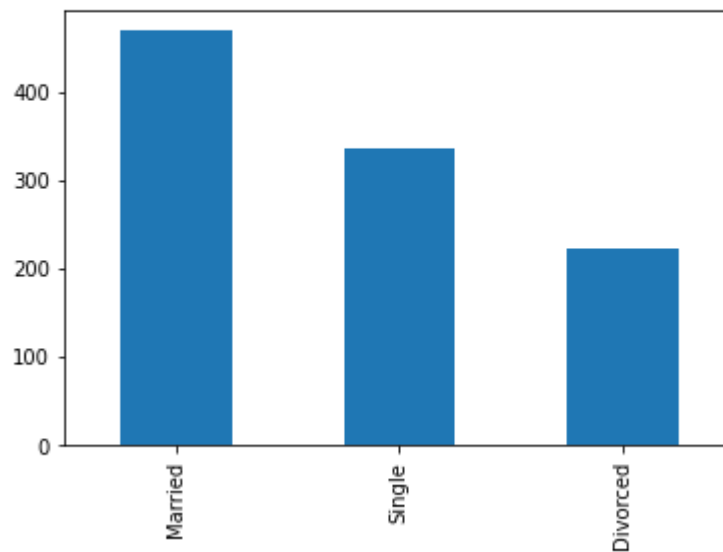
```
[ ] treino['JobSatisfaction'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba7a2c748>



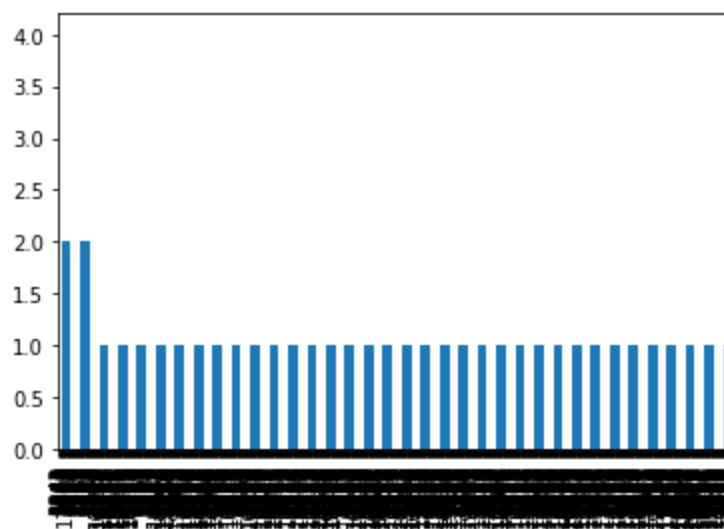
```
[ ] treino['MaritalStatus'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba7a07d68>



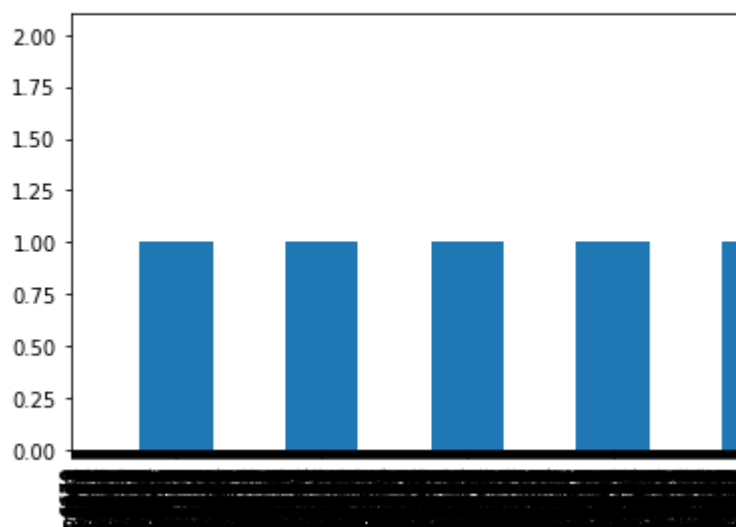
```
[ ] treino['MonthlyIncome'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbb7a74d68>



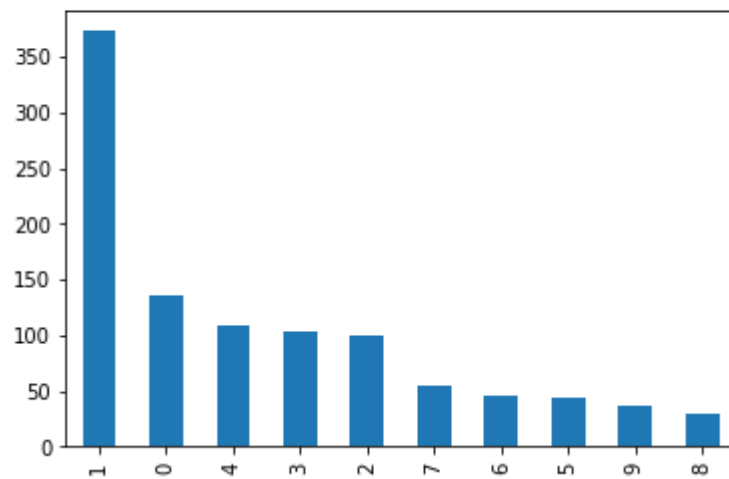
```
[ ] treino['MonthlyRate'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbb6f61080>



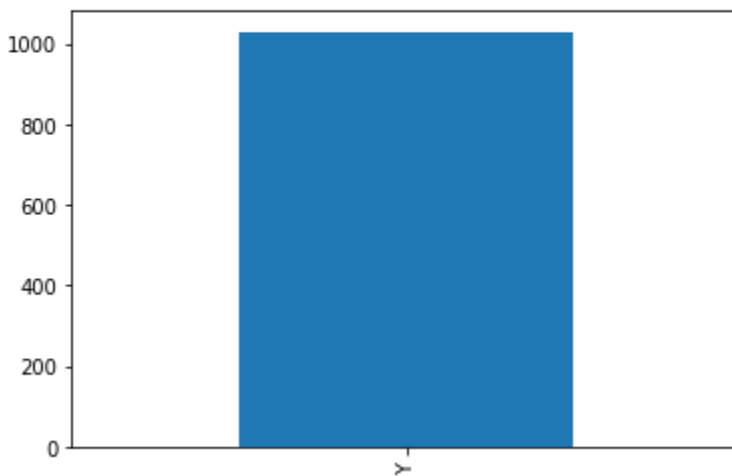
```
[ ] treino['NumCompaniesWorked'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba5268be0>



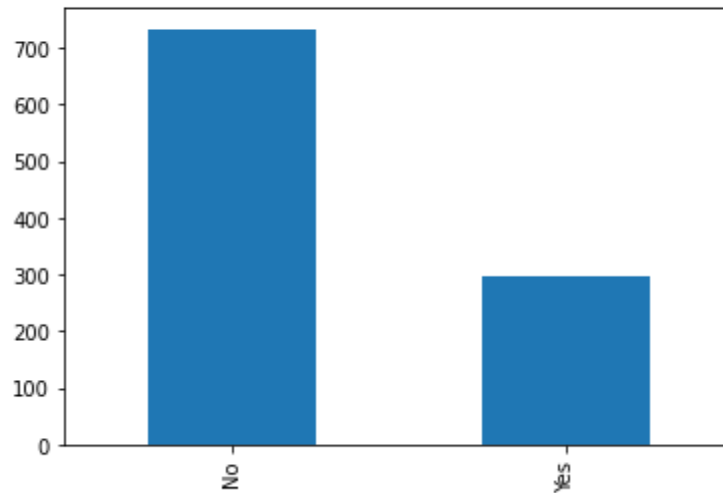
```
▶ treino['Over18'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba52a44e0>



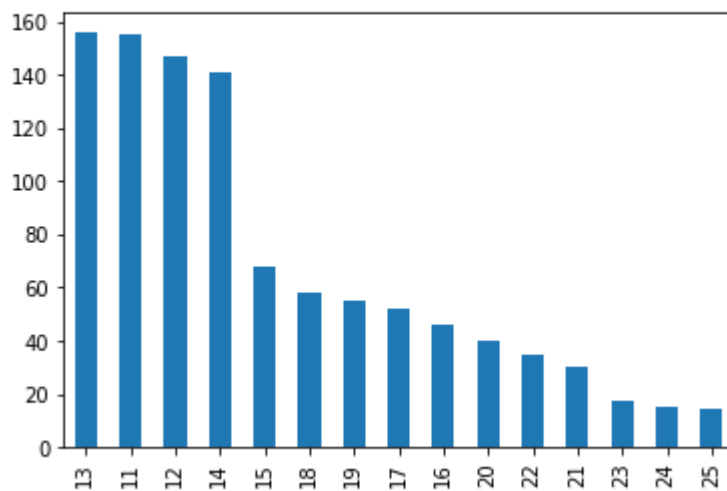
```
▶ treino['OverTime'].value_counts().plot.bar()
```

```
↳ <matplotlib.axes._subplots.AxesSubplot at 0x7fbba4f997f0>
```



```
[ ] treino['PercentSalaryHike'].value_counts().plot.bar()
```

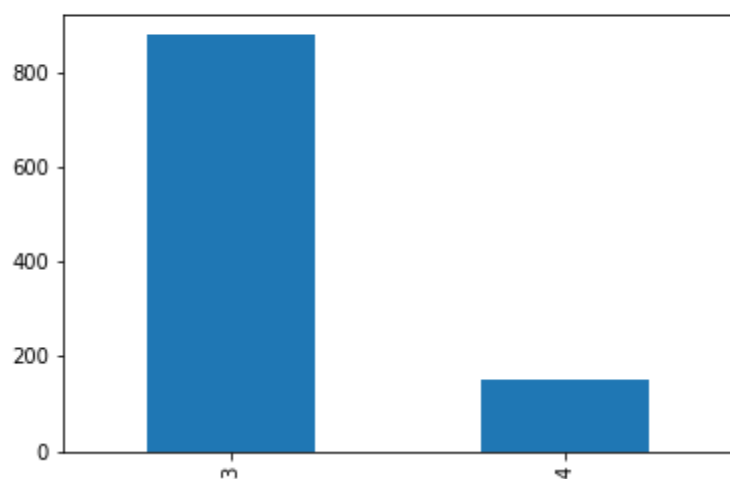
```
↳ <matplotlib.axes._subplots.AxesSubplot at 0x7fbba52bddd8>
```





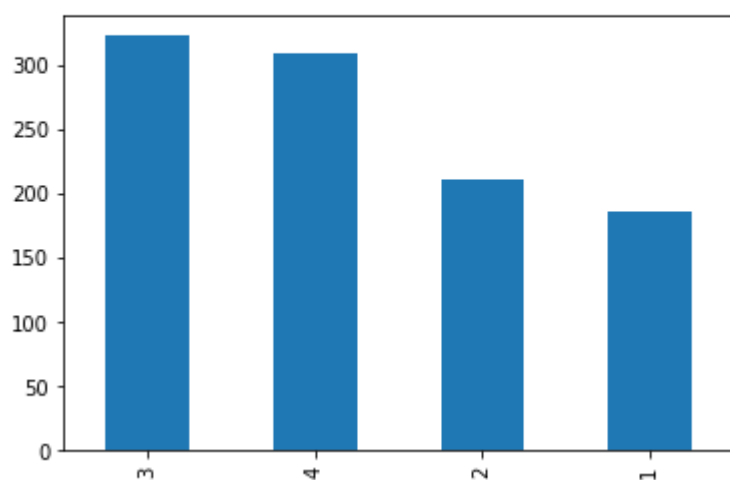
```
[ ] treino['PerformanceRating'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbb52bdb00>



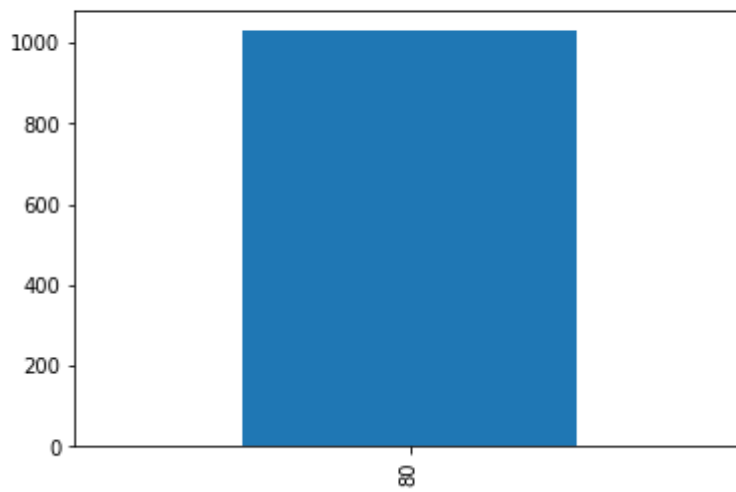
```
[ ] treino['RelationshipSatisfaction'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbb54e82748>



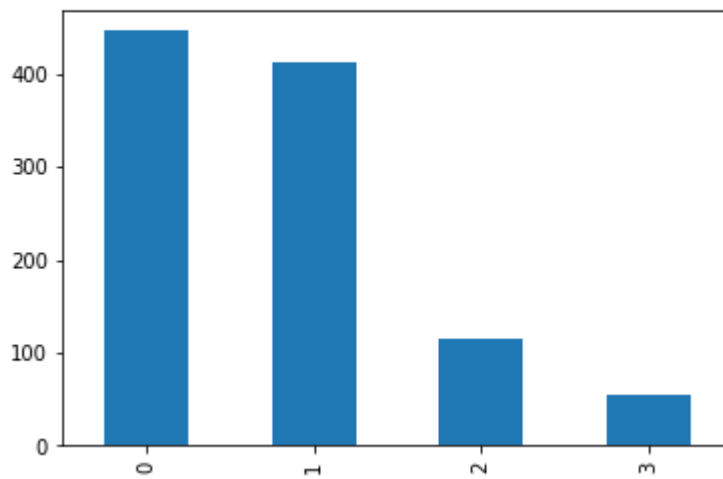
```
[ ] treino['StandardHours'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba4e0a8d0>



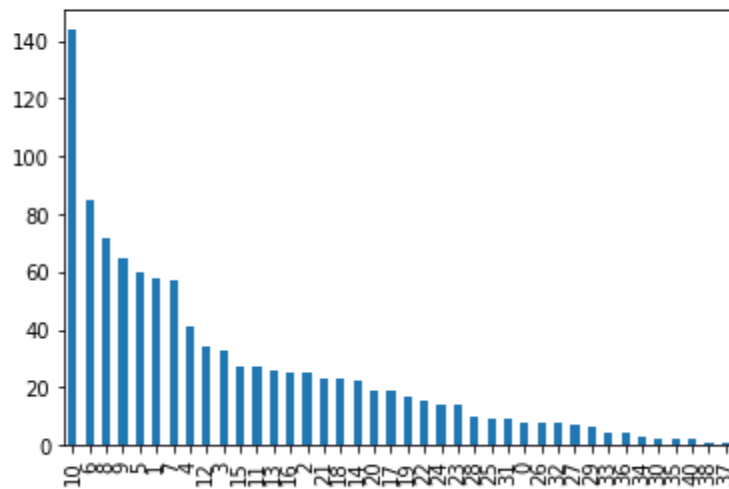
```
[ ] treino['StockOptionLevel'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba4d56208>



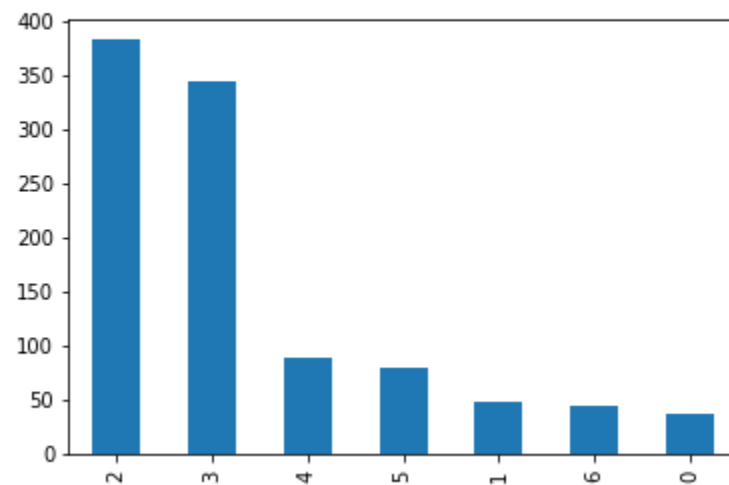
```
[ ] treino['TotalWorkingYears'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba4d4f5c0>



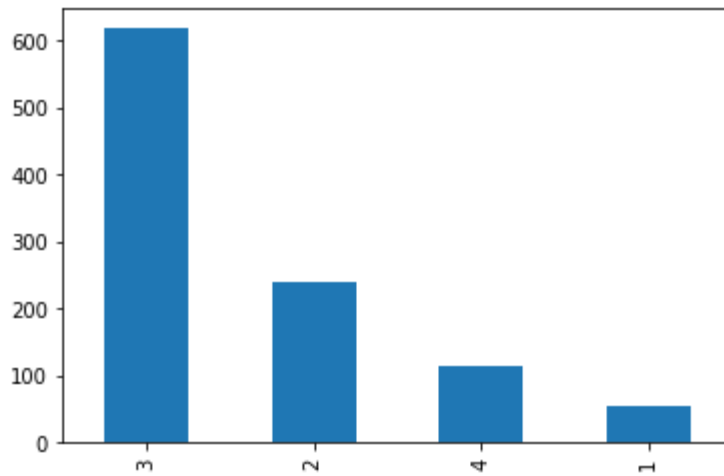
▶ treino['TrainingTimesLastYear'].value\_counts().plot.bar()

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba4be9cc0>



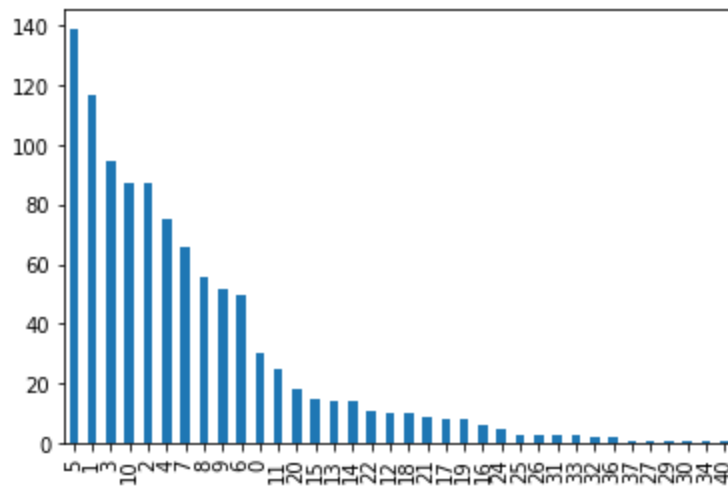
```
[ ] treino['WorkLifeBalance'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbb4be9828>



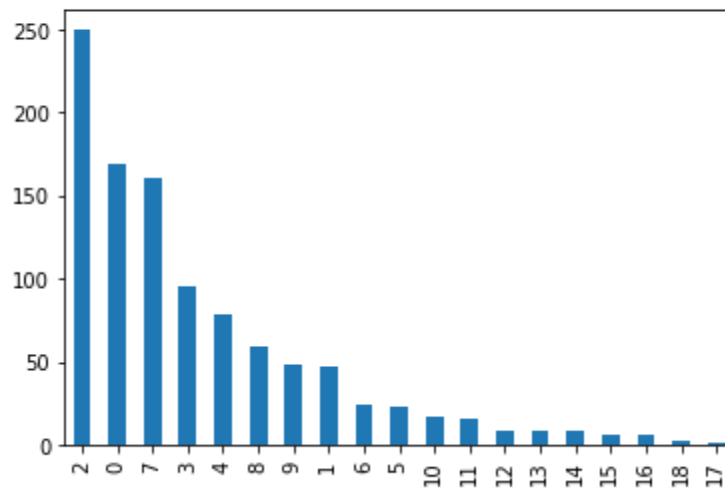
▶ treino['YearsAtCompany'].value\_counts().plot.bar()

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbb4be9c88>



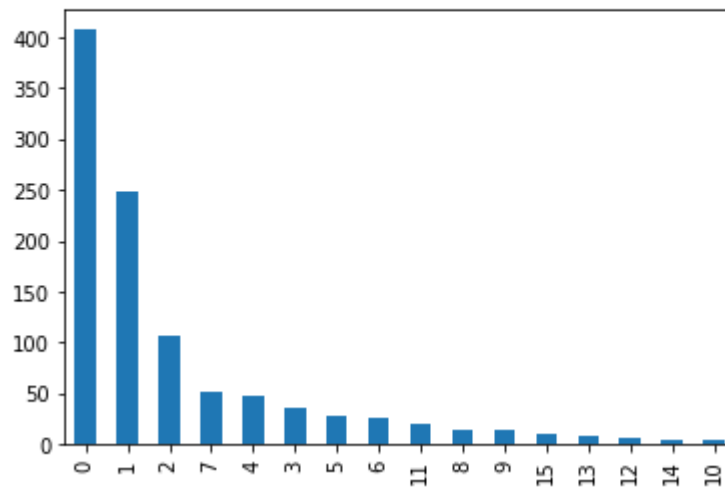
```
[ ] treino['YearsInCurrentRole'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba4b36d68>



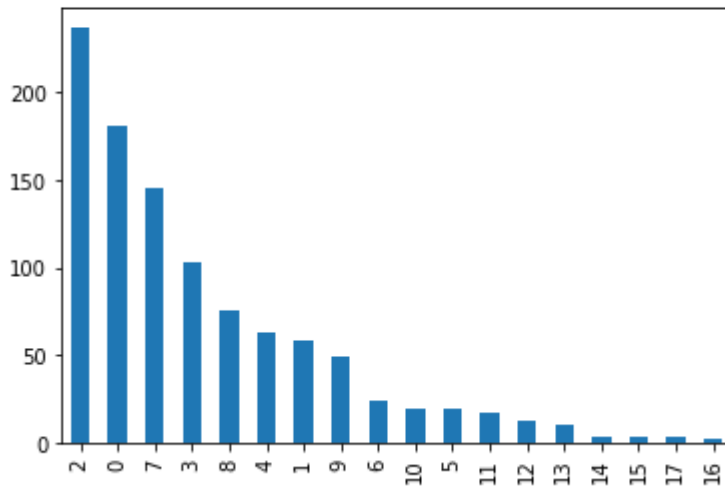
```
[ ] treino['YearsSinceLastPromotion'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba4a24e80>



```
[ ] treino['YearsWithCurrManager'].value_counts().plot.bar()
```

↳ <matplotlib.axes.\_subplots.AxesSubplot at 0x7fbba49943c8>



As seguintes conclusões puderam ser obtidas através da análise exploratória dos dados:

1. As variáveis *EmployeeCount*, *Over18* e *StandardHours* têm apenas um valor único cada, portanto devem ser removida da base de dados utilizada para a aplicação dos algoritmos, uma vez que estas variáveis não serão relevantes.
2. A variável *EmployeeNumber* tem um valor único para cada registro da base, portanto também deve ser removida da base de dados utilizada para a aplicação dos algoritmos.
3. Os campos *Attrition*, *BusinessTravel*, *Department*, *EducationField*, *Gender*, *JobRole*, *MaritalStatus* e *OverTime* são categóricos, e será necessário criar campos numéricos para representar os dados contidos neles.

## 5. Tratamento da base de dados

Conforme indicado na seção anterior, alguns campos foram removidos da base, pois não seriam relevantes para os algoritmos.

### Remoção de campos desnecessários

```
[ ] treino.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis="columns", inplace=True)
treino
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Env
0	42	Yes	Travel_Frequently	933	Research & Development	19	3	Medical	
1	38	No	Travel_Rarely	330	Research & Development	17	1	Life Sciences	
2	43	No	Travel_Frequently	559	Research & Development	10	4	Life Sciences	

Os campos categóricos foram transformados em campos numéricos:

## Transformação de campos categóricos em numéricos

```
[ ] treino["Attrition"] = treino["Attrition"].astype('category')
    treino["AttritionNum"] = treino["Attrition"].cat.codes
```

```
[ ] treino["BusinessTravel"] = treino["BusinessTravel"].astype('category')
    treino["BusinessTravelNum"] = treino["BusinessTravel"].cat.codes
```

```
[ ] treino["Department"] = treino["Department"].astype('category')
    treino["DepartmentNum"] = treino["Department"].cat.codes
```

```
[ ] treino["EducationField"] = treino["EducationField"].astype('category')
    treino["EducationFieldNum"] = treino["EducationField"].cat.codes
```

```
[ ] treino["Gender"] = treino["Gender"].astype('category')
    treino["GenderNum"] = treino["Gender"].cat.codes
```

```
[ ] treino["JobRole"] = treino["JobRole"].astype('category')
    treino["JobRoleNum"] = treino["JobRole"].cat.codes
```

```
[ ] treino["MaritalStatus"] = treino["MaritalStatus"].astype('category')
    treino["MaritalStatusNum"] = treino["MaritalStatus"].cat.codes
```

```
[ ] treino["OverTime"] = treino["OverTime"].astype('category')
    treino["OverTimeNum"] = treino["OverTime"].cat.codes
```



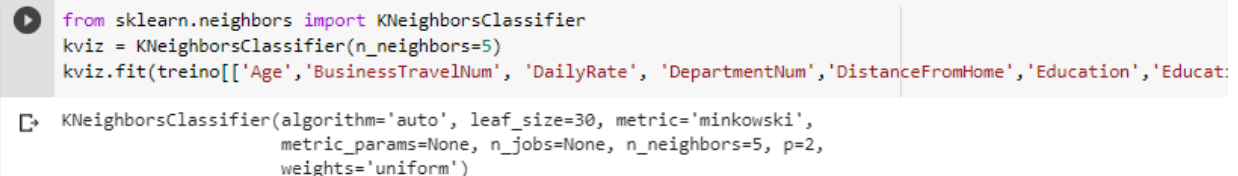
## 6. Aplicação do algoritmo KNN

Conforme mencionado na seção 3, o algoritmo KNN foi selecionado para identificar funcionários mais propícios a deixar a empresa. Desta forma, o algoritmo será utilizado para prever a variável *Attrition*, que indica se um funcionário deixou ou não a empresa.

Como a variável *Attrition* original é categórica, foi criada a variável numérica *AttritionNum* para representar os mesmos dados da mesma. Portanto o modelo a ser criado objetivará prever valores da variável *AttritionNum*.

O seguinte comando foi utilizado para a criação do modelo:

KNN: Criação do modelo



```
from sklearn.neighbors import KNeighborsClassifier
kviz = KNeighborsClassifier(n_neighbors=5)
kviz.fit(treino[['Age', 'BusinessTravelNum', 'DailyRate', 'DepartmentNum', 'DistanceFromHome', 'Education', 'Educat:

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform')
```

Como o comando completo não é exibido por inteiro na imagem acima, segue abaixo o comando executado:

```
from sklearn.neighbors import KNeighborsClassifier
kviz = KNeighborsClassifier(n_neighbors=5)
kviz.fit(treino[['Age', 'BusinessTravelNum', 'DailyRate',
'DepartmentNum', 'DistanceFromHome', 'Education', 'EducationFieldNum', 'EnvironmentSatisfaction', 'GenderNum', 'HourlyRate', 'JobInvolvement', 'JobLevel', 'JobRoleNum', 'JobSatisfaction', 'MaritalStatusNum', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'OverTimeNum', 'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion', 'YearsWithCurrManager']],
treino['AttritionNum']))
```

O modelo foi criado utilizando-se de todas as variáveis disponíveis e o *n\_neighbors* (número de vizinhos) utilizado foi o padrão (5).

Após a criação do modelo, partiu-se para a condução de um teste do mesmo. Para tanto, alguns comandos foram executados para importar e tratar a base de teste:

## KNN: Importação dos dados de teste

```
[ ] teste = pd.read_csv('/content/drive/My Drive/PROJ/dados/ibm_teste.csv')
teste
```

```
↳
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Ed
0	52	No	Non-Travel	585	Sales	29	
1	35	No	Travel Rarelv	1137	Research &	21	

```
[ ] teste.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis="columns", inplace=True)
```

## KNN: Transformação de campos categóricos em numéricos

```
[ ] teste["Attrition"] = teste["Attrition"].astype('category')
teste["AttritionNum"] = teste["Attrition"].cat.codes
```

```
[ ] teste["BusinessTravel"] = teste["BusinessTravel"].astype('category')
teste["BusinessTravelNum"] = teste["BusinessTravel"].cat.codes
```

```
[ ] teste["Department"] = teste["Department"].astype('category')
teste["DepartmentNum"] = teste["Department"].cat.codes
```

```
[ ] teste["EducationField"] = teste["EducationField"].astype('category')
teste["EducationFieldNum"] = teste["EducationField"].cat.codes
```

```
[ ] teste["Gender"] = teste["Gender"].astype('category')
teste["GenderNum"] = teste["Gender"].cat.codes
```

```
[ ] teste["JobRole"] = teste["JobRole"].astype('category')
teste["JobRoleNum"] = teste["JobRole"].cat.codes
```

```
[ ] teste["MaritalStatus"] = teste["MaritalStatus"].astype('category')
teste["MaritalStatusNum"] = teste["MaritalStatus"].cat.codes
```

```
[ ] teste["OverTime"] = teste["OverTime"].astype('category')
teste["OverTimeNum"] = teste["OverTime"].cat.codes
```

Após o tratamento dos dados de teste, foi criada a variável para armazenar os valores preditos da variável *AttritionNum* utilizando-se do modelo criado.

### KNN: Criação da variável de previsões

[illegible]

A seguir foi criada a matriz de confusão, comparando os valores da variável *AttritionNum* preditos pelo modelo criado, e os valores da variável *AttritionNum* reais contidos na base de teste.

### KNN: Matriz de confusão

```
[ ] matriz = pd.crosstab(teste['AttritionNum'] , previsoes, rownames=['Real'], colnames=['Predito'])
matriz
```

	Predito	
Real	0	1
0	365	11
1	55	10

Por fim, foi calculada a acurácia do modelo:

## KNN: Acurácia

```
[ ] from sklearn import metrics
    print(metrics.classification_report(teste['AttritionNum'],previsoes))
```

```

      precision    recall  f1-score   support

     0       0.87      0.97      0.92      376
     1       0.48      0.15      0.23       65

 accuracy          0.85      441
 macro avg       0.67      0.56      0.57      441
 weighted avg    0.81      0.85      0.82      441
```

Pode-se concluir que a acurácia do modelo criado é de 85%.

Modelos com diferentes valores para o parâmetro *n\_neighbors* também foram testados, porém nenhum superou a acurácia de 85%. Para registro, segue uma tabela apresentando a acurácia dos vários modelos criados:

n_neighbors	Acurácia
2	84%
3	81%
4	85%
5	85%
6	85%

## 7. Aplicação do algoritmo de rede neural

Para responder ao segundo problema apresentado na seção 3, utilizaremos um algoritmo de rede neural.

O objetivo será determinar se um funcionário é propício a deixar a empresa baseando-se exclusivamente em poucas variáveis categóricas, que poderiam ser facilmente obtidas por uma empresa através de uma enquête.

As variáveis utilizadas são as seguintes:

1. **Education:** representa o nível de educação formal do funcionário. Pode receber os seguintes valores:
  - 1 'Below College' (Sem Ensino Superior)
  - 2 'College' (Ensino Superior inferior a Bacharelado)
  - 3 'Bachelor' (Bacharelado)
  - 4 'Master' (Mestrado)
  - 5 'Doctor' (Doutorado)
2. **EnvironmentSatisfaction:** representa o nível de satisfação com o ambiente declarado pelo funcionário. Pode receber os seguintes valores:
  - 1 'Low' (Baixo)
  - 2 'Medium' (Médio)
  - 3 'High' (Alto)
  - 4 'Very High' (Muito alto)
3. **JobSatisfaction:** representa o nível de satisfação com o trabalho declarado pelo funcionário. Pode receber os seguintes valores:
  - 1 'Low' (Baixo)
  - 2 'Medium' (Médio)
  - 3 'High' (Alto)
  - 4 'Very High' (Muito alto)
4. **WorkLifeBalance:** representa o nível de equilíbrio entre a vida pessoal e profissional declarado pelo funcionário. Pode receber os seguintes valores:
  - 1 'Bad' (Ruim)
  - 2 'Good' (Bom)
  - 3 'Better' (Muito bom)

- 4 'Best' (Excelente)

As variáveis apresentadas acima serão utilizadas para determinar a propensão de um funcionário de deixar a empresa. O dado referente ao funcionário ter deixado ou não a empresa encontra-se na base de treino na variável *AttritionNum*.

Uma vez que o problema a ser respondido foi apresentado, iniciamos criando as bases de treino e teste.

#### Redes Neurais: Criação das bases de teste e treino

```
[203] treino2 = treino[['Education', 'JobSatisfaction', 'WorkLifeBalance', 'EnvironmentSatisfaction', 'AttritionNum']]
      teste2 = treino[['Education', 'JobSatisfaction', 'WorkLifeBalance', 'EnvironmentSatisfaction']]
      teste2
```

	Education	JobSatisfaction	WorkLifeBalance	EnvironmentSatisfaction
0	3	3	3	3
1	1	3	2	3
2	4	3	4	3
-	-	-	-	-

Em seguida, criamos um modelo de rede neural utilizando o *tensorflow*:

```
[200] import tensorflow
      from tensorflow import keras
      rn = keras.Sequential([
          keras.layers.Input(len(treino2.columns)-1),
          keras.layers.Dense(3, 'relu'),
          keras.layers.Dense(2, 'relu'),
          keras.layers.Dense(1, 'sigmoid')
      ])
      rn.compile(optimizer='adam', loss='binary_crossentropy')
      rn.fit(treino2[['Education', 'JobSatisfaction', 'WorkLifeBalance', 'EnvironmentSatisfaction']], treino2['AttritionNum'])
      previsoes2 = rn.predict(teste[['Education', 'JobSatisfaction', 'WorkLifeBalance', 'EnvironmentSatisfaction']])
      previsoes2
```

```
33/33 [=====] - 0s 1ms/step - loss: 0.6982
array([[0.49217528],
       [0.49217528],
       [0.49217528],
       [0.49217528],
       [0.49217528],
       [0.49217528],
       [0.54468215],
       [0.49217528],
```

Após criar o modelo, criamos uma função de ativação com o limiar de 0.50.

## Redes Neurais: Criação da função de ativação

```
[213] def ativacao(v):  
    limiar = 0.50  
    if (v > limiar):  
        return 1  
    else:  
        return 0  
  
    for previsao2 in previsoes2:  
        print(ativacao(previsao2))
```

```
0  
0  
0  
0
```

Por fim, copiamos os valores de saída do comando apresentado anteriormente e os inserimos na base de teste. Criamos também uma coluna de Resultado, que compara o valor de *AttritionNum* predito pelo modelo ao valor de *AttritionNum* real. Este processo foi feito no Excel.

H	AI	AJ	AK	AL	
Since	YearsWith	AttritionNumPredito	AttritionNumReal	Resultado	
0	0	0	0	Certo	
0	7	0	0	Certo	
0	0	0	1	Errado	
0	7	0	0	Certo	
10	1	0	0	Certo	
0	7	0	0	Certo	
0	0	1	0	Errado	
2	2	0	0	Certo	

A acurácia do modelo é apresentada na tabela abaixo:

Resultado	Contagem	% Acurácia
Certo	345	78.23%
Errado	96	21.77%
Total	441	