PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA

Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Eliane Sumire Ishida

VELOCIDADE DE ADOÇÃO DE UM ANIMAL DE ESTIMAÇÃO

Brasília
2021

Eliane Sumire Ishida

VELOCIDADE DE ADOÇÃO DE UM ANIMAL DE ESTIMAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Brasília

2021

SUMÁRIO

1. Introdução

1.1. Contextualização

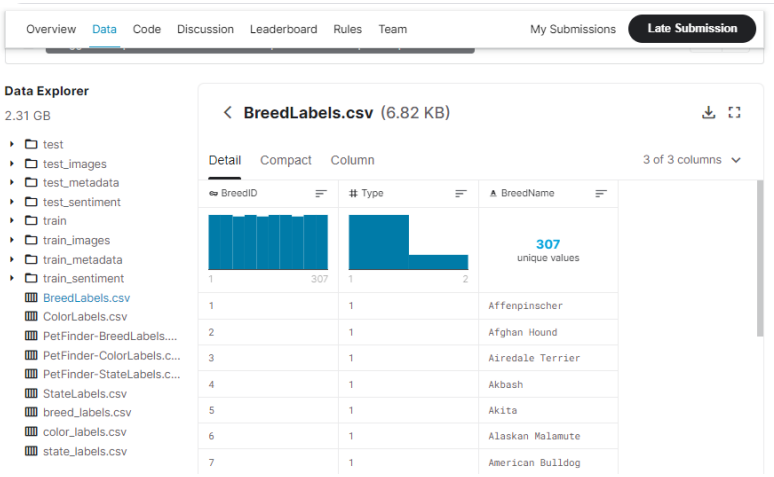
No momento atual de pandemia em que o distanciamento social é necessário para combate à corona vírus acariciar um animal de estimação talvez seja um dos melhores antídotos. Existem pesquisas e estudos que comprovam que esses bichinhos são responsáveis pela sensação de bem-estar e disposição, além de manter a saúde mental dos donos. Ter um bichinho de estimação contribuiu para diminuir o estresse mental de algumas pessoas durante a pandemia. É o que indica o estudo realizado com mais de 5.000 voluntários por umas das principais universidade do Reino Unido, University of York, na Inglaterra, e que foi publicado pelo site plosone. A pesquisa foi realizada entre abril a junho de 2020 e os pesquisadores constataram que aproximadamente 90% dos entrevistados tinham pelo menos um animal de estimação. A maioria dos participantes da pesquisa revelou que ter um animal foi uma fonte de apoio para lidar emocionalmente com o isolamento social. Além da pesquisa dessa universidade, existem outros relatos e notícias disponíveis em sites como Forbes, Galileu, Fundation Affinity, Universidade Federal de Uberlândia falando quanto é benéfico ter o um animal de estimação nesta pandemia.

Essa pequena introdução foi apenas para demonstrar como os animais podem fazer a diferença na nossa vida e também por ser apaixonada por esses bichinhos, então, segui com a indicação da lista disponível de *dataset's* e desenvolvi este trabalho de conclusão da pós-graduação ligada ao tema *Pets*. Neste projeto, tentaremos prever a velocidade de adoção do animal, tendo como base as características dos animais como: a idade, cor, tamanho, condição de saúde e outros. Então, o trabalho foi estruturado em etapas, onde na primeira etapa consiste em *Coletar Dados* para termos todas informações necessárias para o trabalho, a segunda etapa consiste em *Tratar e Preparar Dados*, a terceira etapa é a *Análise Exploratório dos Dados* através de técnicas gráficas de exploração de dados, a quarta etapa é a de *Treinar Modelos de Machine Learning* e a última etapa é a apresentar é *Apresentar o Resultado* do trabalho proposto.

1.2. O problema proposto

No mundo existem diversos animais abandonados e que acabam sendo levados para associações voluntárias, abrigos ou zoonoses, e que infelizmente as vezes demoram para serem adotados ou não conseguem encontrar um lar adotivo.

Os *dataset's* foram fornecidos pela plataforma da Petfinder.my que é dedicada ao bem-estar de animais de estimação na Malásia desde de 2008 e que possui um banco de dados com mais de 150.000 animais. A plataforma possui várias fotos e vídeos com texto descritivo conforme com as características do animal e em certos casos para adotar podem ter que pagar uma taxa de contribuição. Esses conjuntos de dados forma disponibilizados na plataforma do Kaggle acerca 2 anos atrás e para esse projeto serão utilizados os arquivos com extensão csv que são voltadas as características do animal de estimação de cão e gato, só constam esses 2 tipos animais para o trabalho, mas na plataforma existem outros tipos de animais para adoção.



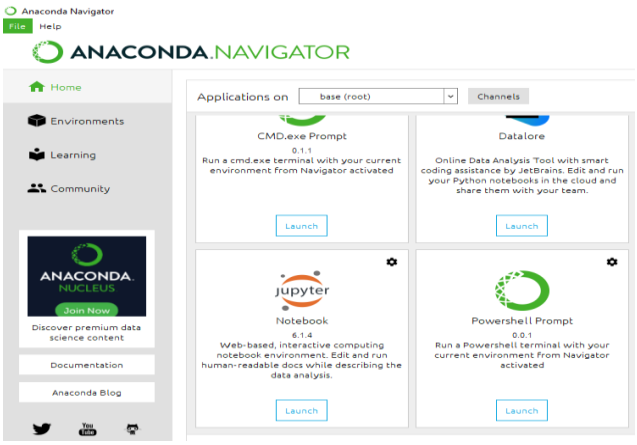
Fonte: autor

Descrição dos arquivos:

- **train.csv** - dados tabulares / de texto para o conjunto de treinamento.
- **breed_labels.csv** - Contém tipo e nome da raça para cada BreedID. Tipo igual 1 é cachorro, 2 é gato.
- **color_labels.csv** - contém o nome da cor para cada ColorID.
- **state_labels.csv** - contém o nome do estado para cada StateID.

2. Coleta de Dados

Conforme citado anteriormente, o conjunto de dados foi obtido do sítio Kaggle e para proposta desse projeto os *dataset's* foram baixados na máquina local. Utilizaremos a ferramenta *Anaconda Navigator e* através do *Jupyter Notebook (versão 6.1.4)* podemos implementar o código na linguagem *Python*.



Fonte: autor

Leitura do repositório *Kaggle* e criação dos dataset's (train, test state, breed, color):

```
#arquivos
train_dataset_path = "../PUC_TCC/train.csv"
train = pd.read_csv(train_dataset_path)

test_dataset_path = "../PUC_TCC/test.csv"
test = pd.read_csv(test_dataset_path)

state_dataset_path = "../PUC_TCC/state_labels.csv"
state_labels = pd.read_csv(state_dataset_path)

breed_dataset_path = "../PUC_TCC/breed_labels.csv"
breed_labels= pd.read_csv(breed_dataset_path)

color_dataset_path = "../PUC_TCC/color_labels.csv"
color_labels = pd.read_csv(color_dataset_path)
```

Fonte: autor

Dataset: **train.csv**

Contém informações do perfil do tipo do animal: cão e gato. É a nossa base de treino e onde a variável alvo (*target*) é *AdoptionSpeed* que contém os valores de domínios com tempo de adoção:

- ✓ 0 - Pet foi adotado no mesmo dia em que foi listado.
- ✓ 1 - Pet foi adotado entre 1 e 7 dias (1ª semana) após ser listado.
- ✓ 2 - Pet foi adotado entre 8 e 30 dias (1º mês) após ser listado.

- ✓ 3 - O animal de estimação foi adotado entre 31 e 90 dias (2º e 3º mês) após ser listado.
- ✓ 4 - Não há adoção após 100 dias de listagem. (Não há animais de estimação neste conjunto de dados que esperaram entre 90 e 100 dias).

Contém 14.993 registros e 24 atributos do tipo numérico, decimal e descritivo.

```
train.head(2)
```

	Type	Name	Age	Breed1	Breed2	Gender	Color1	Color2	Color3	MaturitySize	...	Health	Quantity	Fee	State	RescuerID	Vide
0	2	Nibble	3	299	0	1	1	7	0	1	...	1	1	100	41326	8480853f516546f6cf33aa88cd76c379	
1	2	No Name Yet	1	265	0	1	1	2	0	2	...	1	1	0	41401	3082c7125d8fb66f7dd4bff4192c8b14	

2 rows × 24 columns

Fonte: autor

Layout train.csv:

Nome da coluna/campo	**Descrição**	**Tipo**
Type	Tipo de animal (1 = Cachorro, 2 = Gato).	int64
Name	Nome do animal de estimação (vazio se não for nomeado).	object
Age	Idade do animal de estimação quando listado, em meses.	int64
Breed1	ID da raça primária de animal de estimação (consulte o dicionário BreedLabels.csv).	int64
Breed2	ID da raça secundária de animal de estimação, se o animal for de raça mista (consulte o dicionário BreedLabels.csv).	int64
Gender	Sexo do animal de estimação (1 = Masculino, 2 = Feminino, 3 = Misto, se o perfil representar um grupo de animais de estimação).	int64
Color1	Cor 1 do animal de estimação (consulte o dicionário ColorLabels.csv).	int64
Color2	Cor 2 do animal de estimação (consulte o dicionário ColorLabels.csv).	int64
Color3	Cor 3 do animal de estimação (consulte o dicionário ColorLabels.csv).	int64
MaturitySize	Tamanho na maturidade (1 = Pequeno, 2 = Médio, 3 = Grande, 4 = Extragrande, 0 = Não Especificado).	int64
FurLength	Comprimento do pelo (1 = curto, 2 = médio, 3 = longo, 0 = não especificado).	int64
Vaccinated	O animal de estimação foi vacinado (1 = Sim, 2 = Não, 3 = Não tenho certeza).	int64
Dewormed	O animal de estimação foi desparasitado (1 = Sim, 2 = Não, 3 = Não tenho certeza).	int64
Sterilized	Animal de estimação foi esterilizado / castrado (1 = Sim, 2 = Não, 3 = Não tenho certeza).	int64
Health	Condição de saúde (1 = Saudável, 2 = Lesão leve, 3 = Lesão grave, 0 = Não especificado).	int64
Quantity	Número de animais de estimação representados no perfil.	int64

Fee	Taxa de adoção (0 = grátis). [0..300]	int64
State	ID do estado na Malásia (consulte o dicionário StateLabels).	int64
RescuerID	ID único do salvador.	object
VideoAmt	Total de vídeos enviados para este animal de estimação.	int64
Description	Descrição do perfil do animal de estimação.	object
PetID	ID exclusivo do perfil do animal de estimação.	object
PhotoAmt	Total de fotos enviadas para este animal de estimação.	float64
AdoptionSpeed	Velocidade categórica de adoção. Abaixar é mais rápido. Este é o valor a prever. Veja a seção abaixo para mais informações: 0 - Pet foi adotado no mesmo dia em que foi listado. 1 - Pet foi adotado entre 1 e 7 dias (1ª semana) após ser listado. 2 - Pet foi adotado entre 8 e 30 dias (1º mês) após ser listado. 3 - O animal de estimação foi adotado entre 31 e 90 dias (2º e 3º mês) após ser listado. 4 - Não há adoção após 100 dias de listagem. (Não há animais de estimação neste conjunto de dados que esperaram entre 90 e 100 dias).	int64

Dataset**: state_label.csv**

É a tabela de domínio que contém o nome do estado da Malásia. Faz referência com o *dataset train*, atributo *ID state*.

Possuem 15 registos e 2 atributos do tipo de numérico e descritivo.

```
state_labels.head(2)
```

	StateID	StateName
0	41336	Johor
1	41325	Kedah

Fonte: autor

Layout state_label.csv:

Nome da coluna/campo	**Descrição**	**Tipo**
StateID	ID do estado na Malásia.	int64
StateName	Nome estado ada Malásia.	object

Dataset**: breed_label.csv**

É a tabela de domínio que contém a raça do animal. Faz referência com o *dataset train*, atributo *ID breed*.

Possuem 307 instâncias e 3 atributos do tipo de numérico e descritivo.

```
breed_labels.head(2)
```

	BreedID	Type	BreedName
0	1	1	Affenpinscher
1	2	1	Afghan Hound

Fonte: autor

Layout breed_label.csv:

Nome da coluna/campo	Descrição	Tipo
BreedID	ID da raça.	int64
Type	Tipo de animal (1 = Cachorro, 2 = Gato).	int64
BreedName	Nome da raça.	object

Fonte: autor

Dataset: **color_label.csv**

É a tabela de domínio que contém a raça do animal. Faz referência com o *dataset train*, atributo *ID color*.

Possuem 7 instâncias e 2 atributos do tipo de numérico e descritivo.

```
color_labels.head(2)
```

	ColorID	ColorName
0	1	Black
1	2	Brown

Fonte: autor

Layout color_label.csv:

Nome da coluna/campo	Descrição	Tipo
ColorID	ID da raça.	int64
ColorName	Cor 1 do animal de estimação	object

3. Processamento/Tratamento de Dados

Após a coleta dos dados, agora iremos fazer o processamento e tratamento dos dados com a finalidade de verificar:

- dados omissos ou brancos podem ser tratados inferido valores ou excluídos;

- dados codificados ou despadronizados, deve-se verificar se estão com o mesmo formato ou tipo;
- existência de dados ruidosos ou inconsistentes, inexatos ou incompletos que possam afetar o modelo;
- duplicidade de instâncias e de atributos, como por exemplo, atributo tipo data de nascimento e idade ou valor total e valor do produto, podem remover um desses atributos;
- os atributos irrelevantes que podem ser removidos porque podem causar modelo generalizado, como por exemplo, ID único e campo descritivo são atributos que não serão necessários para análise do modelo e assim evitar assim qualquer tipo de ruído de dados;
- remoção de dados conflitantes, ter 2 instâncias para o mesmo campo chave, deve-se manter apenas a informação mais recente;
- tipo de dados quantitativos ou qualitativos, pois dependendo do tipo é importante para construir o modelo de forma adequada porque os dados podem trazer diferentes informação conforme são interpretados;
- dados redundantes (atributos ou registros) que podem estar inconsistentes, o que podem causar conhecimento falso e aumento de tempo de processamento/execução dos algoritmos;

Considerando as observações acima verificamos que no *dataset train* existem atributos nulos: *Name* e *Description* e atributos irrelevantes tipo ID: *PetID*, *RescueID*. Vamos examiná-los melhor na próxima etapa de *Análise de Dados*.

```
#quantidade de nulos
train.isnull().sum().sort_values(ascending=False)
```

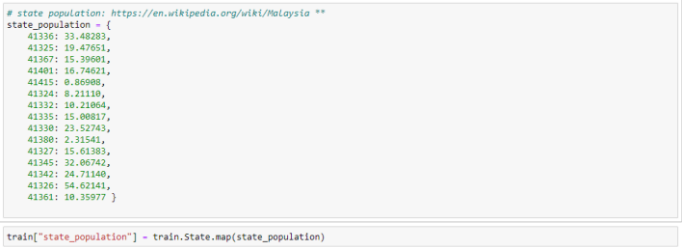
```
Name              1257
Description         12
AdoptionSpeed        0
FurLength            0
Age                  0
Breed1               0
Breed2               0
Gender               0
Color1               0
Color2               0
Color3               0
MaturitySize         0
Vaccinated           0
PhotoAmt             0
Dewormed             0
Sterilized           0
Health               0
Quantity             0
Fee                  0
State                0
RescuerID            0
VideoAmt             0
PetID                0
Type                 0
dtype: int64
```

Fonte: autor

Para enriquecer o *dataset train,* iremos fazer a junção com os outros arquivos trazendo as informações de domínio nome do Estado, Raça e Cor, e que tem como atributos chaves: *StateID, Breed1, Breed2, Color1,Colo2 e Color3*.

Fonte: autor

A título de curiosidade e conhecimentos agregamos também no *dataset train* as informações do tamanho da população dos estados da Malásia (*StateID*) disponível no Wikipédia. Lembrando que nesse *dataset train* consta o atributo *StateID* e fizemos a incorporação das informações através dele.



Fonte: autor

Criamos *flag's* para tratar os campos de cores e raças, se estão ou não com informações. No *dataset* existem 3 atributos referente a cor do animal (Colo1, Color2 e Color3), na qual, apenas o atributo Color1 está preenchido, então cor única no animal, senão, o atributo Color2 também está preenchido, então, o animal tem 2 cores, ou senão, animal possui 3 cores, todos os atributos de cor estão com informações. E para raça existem 2 atributos: Breed1 e Breed2 que entendemos que foi criado para identificar se a raça é pura ou se é mestiça, os nossos conhecidos vira-latas.

```
# Color1, Color2 e Color3
#flag1(1): apenas cor1
train['Color_f1'] = (pd.isnull(train['ColorName_3']) & pd.isnull(train['ColorName_2']) & pd.notnull(train['ColorName_1'])).astype
#flag2(1): cor1 + cor2
train['Color_f2'] = (pd.isnull(train['ColorName_3']) & pd.notnull(train['ColorName_2']) & pd.notnull(train['ColorName_1'])).astyp
#flag3(1): todas 3 cores
train['Color_f3'] = (pd.notnull(train['ColorName_3']) & pd.notnull(train['ColorName_2']) & pd.notnull(train['ColorName_1'])).asty

# Breed1 e Breed2
#flag1(1): apenas breed1
train['Breed_f1'] = (pd.isnull(train['BreedName_2']) & pd.notnull(train['BreedName_1'])).astype(int)
#flag2(1): breed1 + breed2
train['Breed_f2'] = (pd.notnull(train['BreedName_2']) & pd.notnull(train['BreedName_1'])).astype(int)
```

Fonte: autor

Outro tratamento que realizamos foi criar campos descritivo de alguns atributos ou formatação do domínio para facilitar na hora da análise de dados e que poderão ser descartados na hora da criação de modelos: *Type, Age, MaturitySize, FurLength, Vaccinated, Dewormed, Sterilized, Health, Name, Description, PhotoAmt, Video e Fee.*

```
#tratamento atributos
#Type
new_type = pd.Categorical(train["Type"])
new_type = new_type.rename_categories(["Dog","Cat"])
train["Type_Desc"] = new_type

#Gender
new_Gender = pd.Categorical(train["Gender"])
new_Gender = new_Gender.rename_categories(["Macho","Fêmea","Mestiço"])
train["Gender_Desc"] = new_Gender

#MaturitySize
new_MaturitySize = pd.Categorical(train["MaturitySize"])
new_MaturitySize = new_MaturitySize.rename_categories(["Pequeno","Médio","Grande","Extravagante"])
train["MaturitySize_Desc"] = new_MaturitySize

#FurLength
new_FurLength = pd.Categorical(train["FurLength"])
new_FurLength = new_FurLength.rename_categories(["Curto","Médio","Longo"])
train["FurLength_Desc"] = new_FurLength

#Health
new_Health = pd.Categorical(train["Health"])
new_Health = new_Health.rename_categories(["Saudável","Lesão leve","Lesão grave"])
train["Health_Desc"] = new_Health

#Vaccinated
new_Vaccinated = pd.Categorical(train["Vaccinated"])
new_Vaccinated = new_Vaccinated.rename_categories(["Sim","Não","Sem info"])
train["Vaccinated_Desc"] = new_Vaccinated

#Dewormed
new_Dewormed = pd.Categorical(train["Dewormed"])
new_Dewormed = new_Dewormed.rename_categories(["Sim","Não","Sem info"])
train["Dewormed_Desc"] = new_Dewormed
```

Fonte: autor

```
#Sterilized
new_Sterilized = pd.Categorical(train["Sterilized"])
new_Sterilized = new_Sterilized.rename_categories(["Sim","Não","Sem info"])
train["Sterilized_Desc"] = new_Sterilized

#Name (create a flag if the name is missing, with less than two letters)
train['Name_len']=train['Name'].str.len()
train['Name_missing'] =  (pd.isnull(train['Name'])).astype(int)

#Description
train['Description_len']=train['Description'].str.len()

#PhotoAmt - No photo (1)
train['Photo_no'] =  (train['PhotoAmt']==0).astype(int)

#VideoAmt - No Video (1)
train['Video_no'] =  (train['VideoAmt']==0).astype(int)

# Fee - Fee free (1) se for taxa free
train['Fee_free'] =  (train['Fee']==0).astype(int)
```

Fonte: autor

```
# Normalize the Variable Description e Name
train['Description'] =train['Description'].fillna("<MISSING>")
train['Name'] =train['Name'].fillna("<MISSING>")
train['BreedName_1'] =train['BreedName_1'].fillna("<MISSING>")
train['BreedName_2'] =train['BreedName_2'].fillna("<MISSING>")
train['ColorName_2'] =train['ColorName_2'].fillna("<MISSING>")
train['ColorName_3'] =train['ColorName_3'].fillna("<MISSING>")
```

Fonte: autor

4. Análise e Exploração dos Dados

Na fase de análise e exploração dos dados são necessários mais esforços para entender como as variáveis são distribuídas. E para isso foram utilizadas técnicas de visuais de distribuição que favorecem o entendimento e rapidez nos questionamentos. Quais são os tipos de animais de estimação mais adotados, cão ou gato? Qual é a idade dos animais que mais são adotados? Pelos, cores, tamanho que são preferidos na hora da adoção?

Existem várias técnicas de visualização e foram utilizados os gráficos como: *distplot, countplot, catplot e barplot* e que apesar de serem aparentemente simples eles trazem grande benefícios na hora da visualização, tratamento, distribuição e entendimento dos dados.

As variáveis serão analisadas para ver possíveis interações e impacto com a variável *target AdoptionSpeed*.

Estatísticas dos dados:

```
# análise incial - estatíticas
print('Data Statistics:')
train.describe()
```

```
Data Statistics:
```

	Type	Age	Breed1	Breed2	Gender	Color1	Color2	Color3	MaturitySize	FurL
count	14993.000000	14993.000000	14993.000000	14993.000000	14993.000000	14993.000000	14993.000000	14993.000000	14993.000000	14993.0
mean	1.457614	10.452078	265.272594	74.009738	1.776162	2.234176	3.222837	1.882012	1.862002	1.4
std	0.498217	18.155790	60.056818	123.011575	0.681592	1.745225	2.742562	2.984086	0.547959	0.5
min	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	1.0
25%	1.000000	2.000000	265.000000	0.000000	1.000000	1.000000	0.000000	0.000000	2.000000	1.0
50%	1.000000	3.000000	266.000000	0.000000	2.000000	2.000000	2.000000	0.000000	2.000000	1.0
75%	2.000000	12.000000	307.000000	179.000000	2.000000	3.000000	6.000000	5.000000	2.000000	2.0
max	2.000000	255.000000	307.000000	307.000000	3.000000	7.000000	7.000000	7.000000	4.000000	3.0

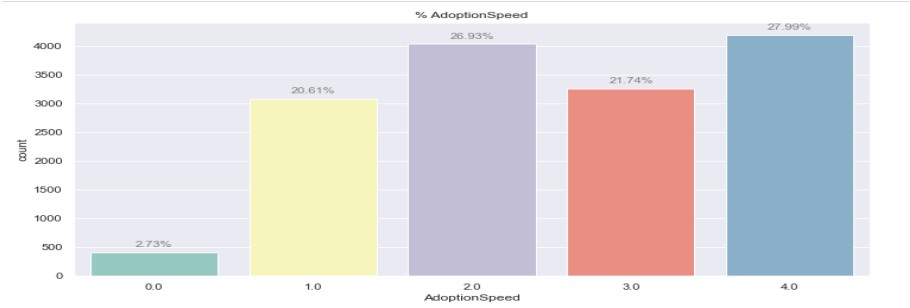
8 rows × 32 columns

Fonte: autor

Visão geral dos atributos *dataset – train:*

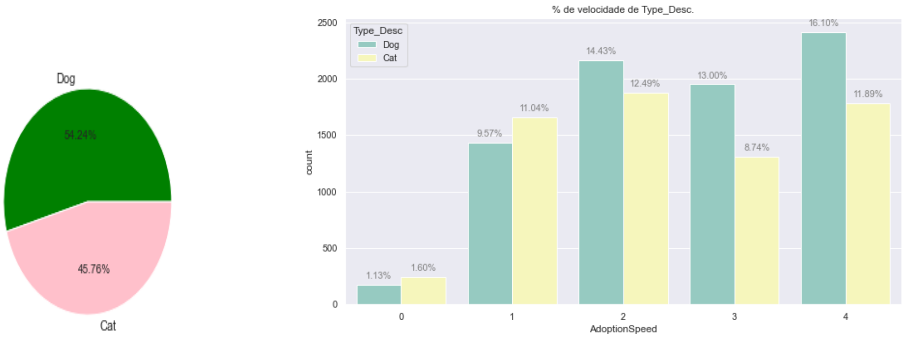
Fonte: autor

- **AdoptionSpeed**: É o atributo alvo do *dataset* – *train,* onde temos a maior quantidade de informação na categoria igual 2 e 4, onde o 2 é a adoção de 8 a 30 dias e o 4 quando a adoção é após 100 dias; categoria igual a 0 é adotado no mesmo dia que é cadastrado na plataforma; categoria igual a 1 equivale a adoção de 1 dia a 7 dias; e a categoria igual a 3 é adoção 31 a 90 dias; observe que não constam informações de adoção de 91 a 100 dias;



Fonte: autor

- **Type**: Tipo do animal, cão ou gato. A média é 1,45 e com isso já sabemos que há mais animais do tipo 1 (cão) do que do tipo 2 (gatos) na base de dados. Existem mais gatos na categoria 0 e 1, ou seja, os gatos são adotados mais rapidamente nos primeiros dias de vida;



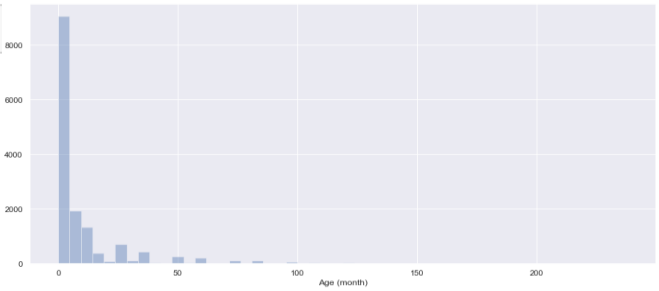
Fonte: autor

- **Age**: Idade (meses) do animal de estimação. Veja na imagem abaixo que a média da idade é 10.45 (meses), entretanto, nesse caso a média não nos dá uma visão da realidade já que a mediana (50%) é 3, ou seja, metade dos animais do *dataset* tinha até 3 meses na adoção. Existem algumas informações meio duvidosas, por exemplo, a idade máxima relatada são 255 meses, ou seja, 21 anos será que existem cão ou gato com essa idade? E a maioria são animais jovens na base de dados. Outro ponto é que o atributo *Age* não corresponde a idade real no momento da adoção e sim a idade quando o animal foi cadastrado na plataforma, por exemplo, **PetID=3422e4906** está com **Age=1** (mês), mas ele foi adotado 2 a 3 meses (**AdoptionSpeed=3**), ou senão, **PetID=fa4314902** com **Age=2 (meses)** e foi adotado no

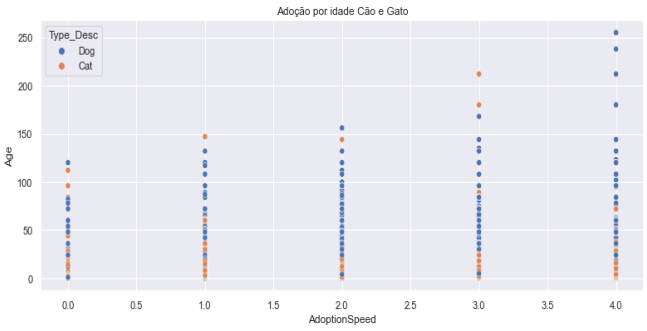
mesmo dia (**AdoptionSpeed=0**) que foi disponibilizado na base de dados da plataforma *Petfinder*;

```
train['Age'].describe()
```

count	14993.000000
mean	10.452078
std	18.155790
min	0.000000
25%	2.000000
50%	3.000000
75%	12.000000
max	255.000000



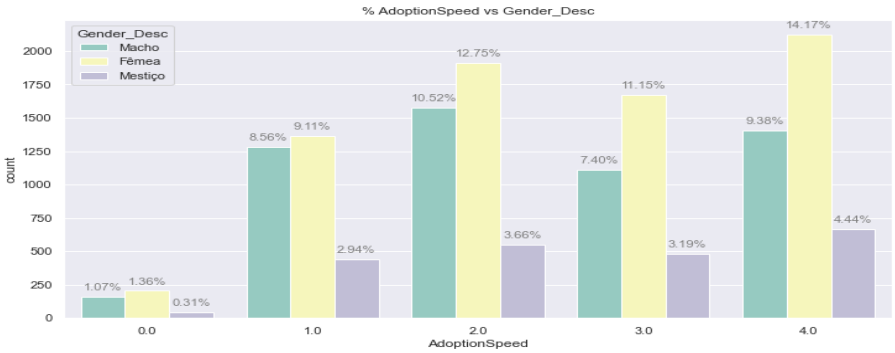
Fonte: autor



	PetID	Type	AdoptionSpeed	Age
1	3422e4906	1	3	1
2	850a43f90	1	2	1
5	97aa9eeac	1	3	0
6	c06d167ca	2	1	2
9	aaedd873d	1	2	2
...	...	...	...	...
14979	f26e3e182	2	2	1
14982	59ee91cf8	2	1	2
14983	fa4314902	2	0	2
14984	a12589a09	1	0	1
14987	2b4003be8	2	2	1

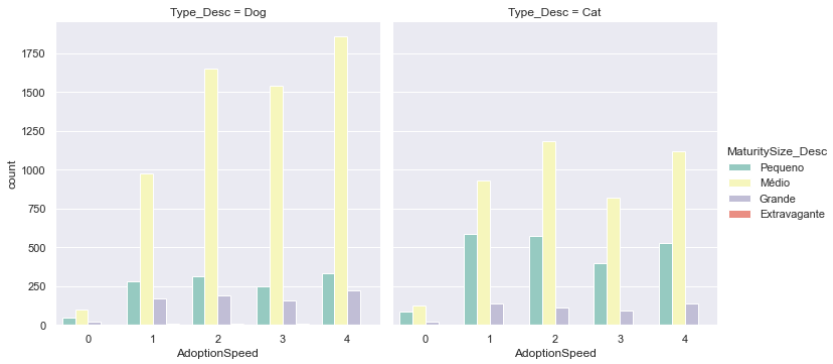
Fonte: autor

- **Gender**: As fêmeas são as preferidas para adoção tanto para cão quanto para gato;



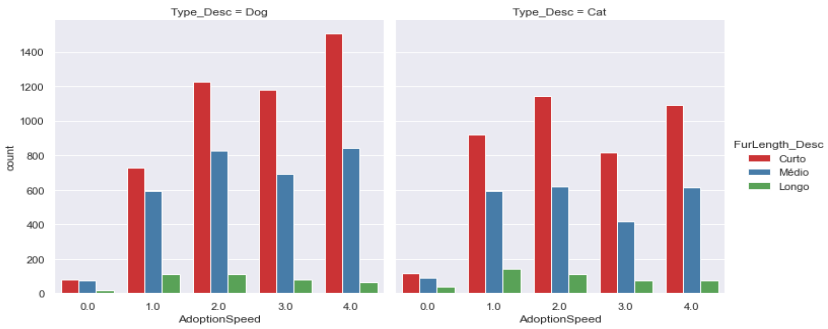
Fonte: autor

- **MaturitySize**: A maioria prefere adotar os animais com o porte médio tanto para cão como para os gatos. E também esse porte médio é o que tendem a serem adotados mais rápido, principalmente os cachorros;



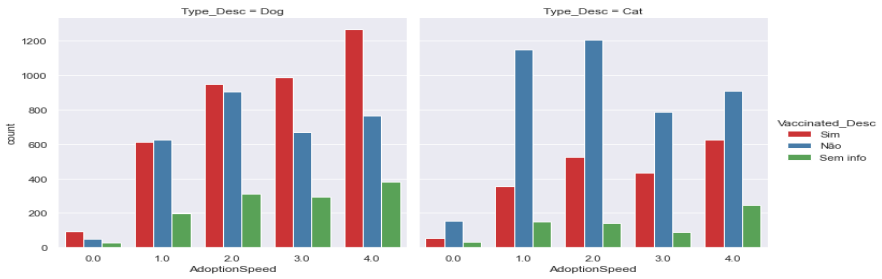
Fonte: autor

- **FurLength**: Já para o comprimento dos pelos de gato e cão, o curto tem mais chance de ser adotado mais rápido;



Fonte: autor

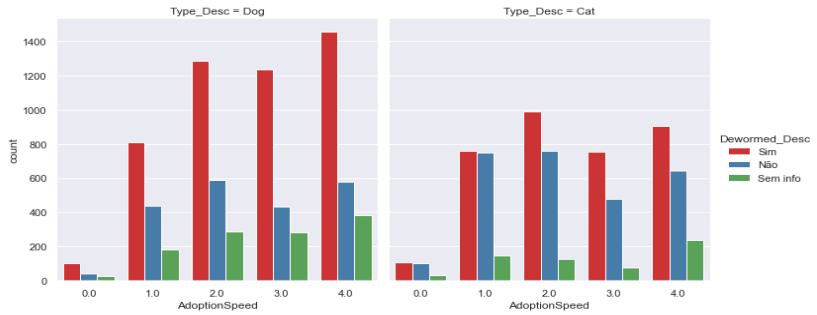
- **Vaccinated**: A maioria dos cachorros são adotados vacinados e para os gatos a vacinação não interfere na sua adoção;



Fonte: autor

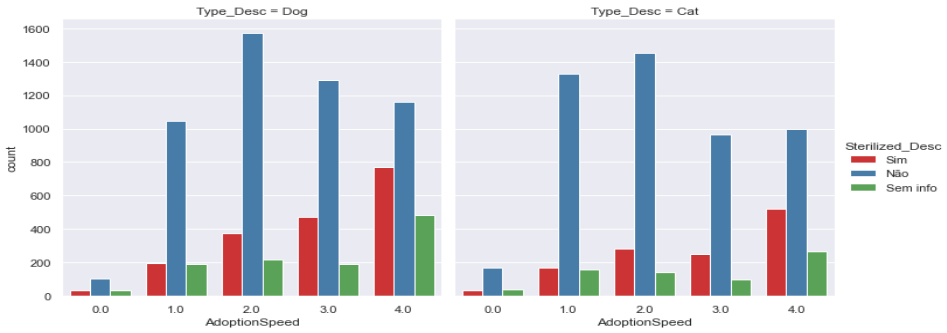
- **Dewormed**: A grande maioria dos animais estão desparasitados para adoção, os cachorros desde pequenos são tratados, os gatos estão

mais equilibrados nas 2 primeiras categorias e a partir da categoria 2 eles são mais vermifugados;

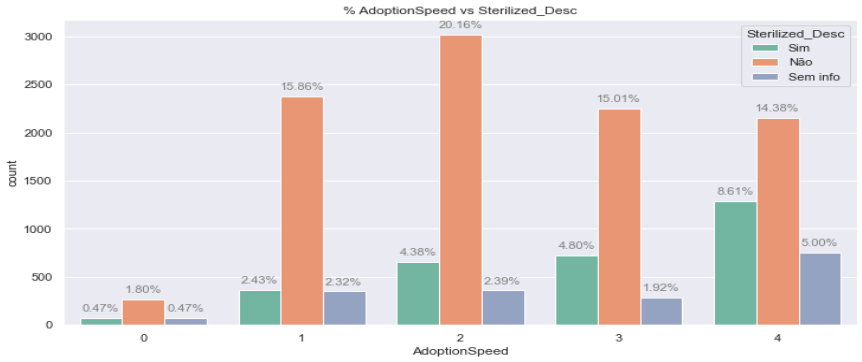


Fonte: autor

- **Sterilized**: Ser castrado não interfere na adoção de gatos ou cachorros;

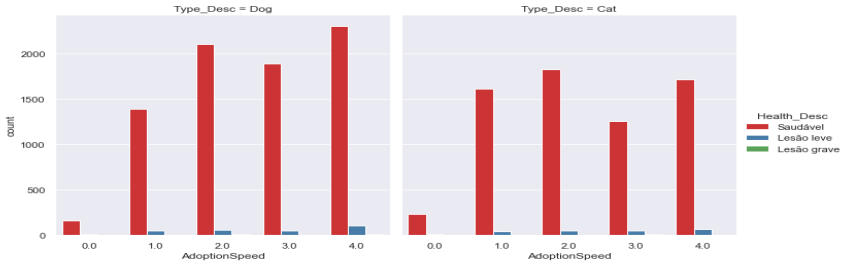


Fonte: autor

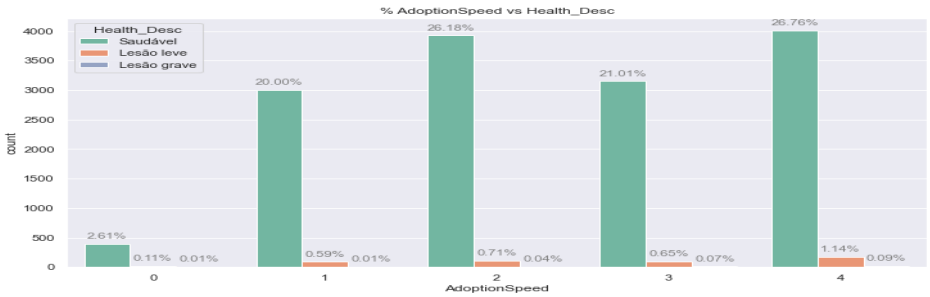


Fonte: autor

- **Heath**: A maioria dos animais são adotados saudáveis, mas isso não impede que os lesionados sejam adotados também; e é um alívio saber que na base existam poucos casos de animais com problema de saúde;

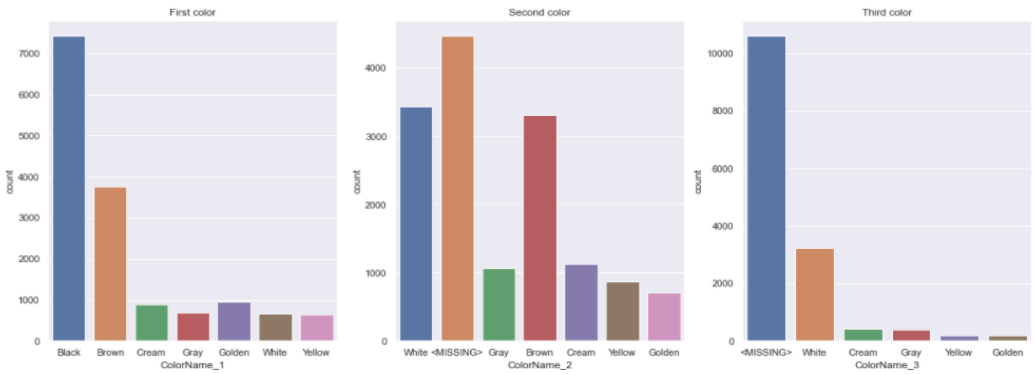


Fonte: autor

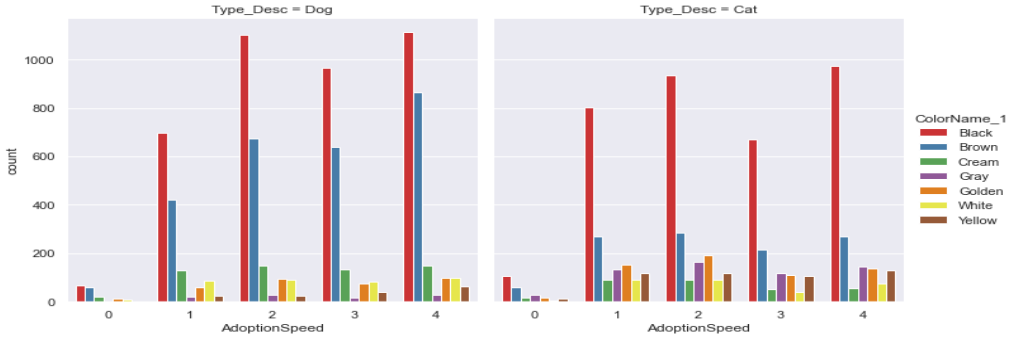


Fonte: autor

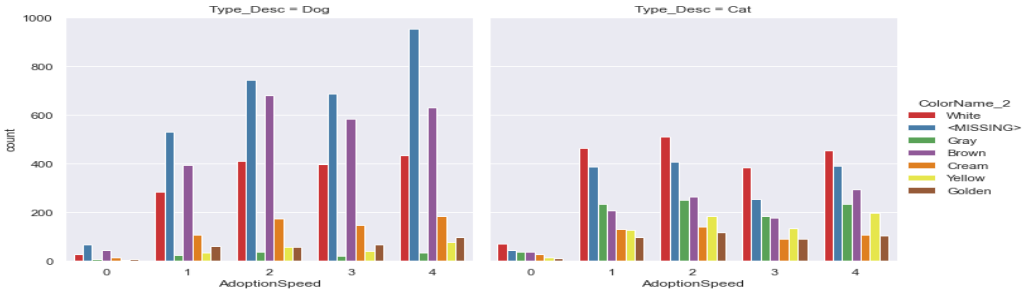
- **Color1, Color2 e Color3:** A cor preta, marrom e branco são as predominantes nesses 3 atributos e também na escolha na hora adoção para dos cães e gatos;



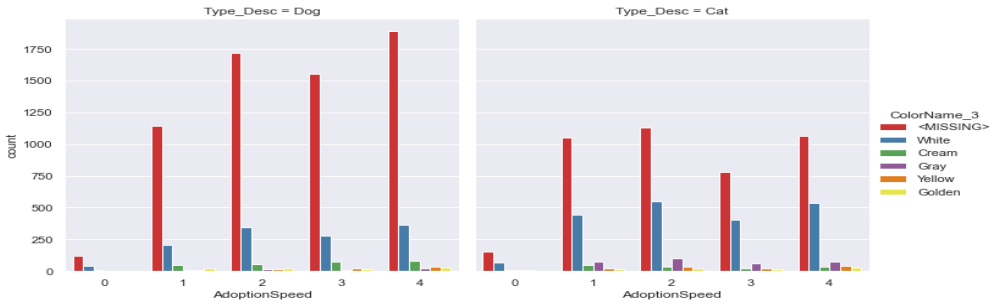
Fonte: autor



Fonte: autor

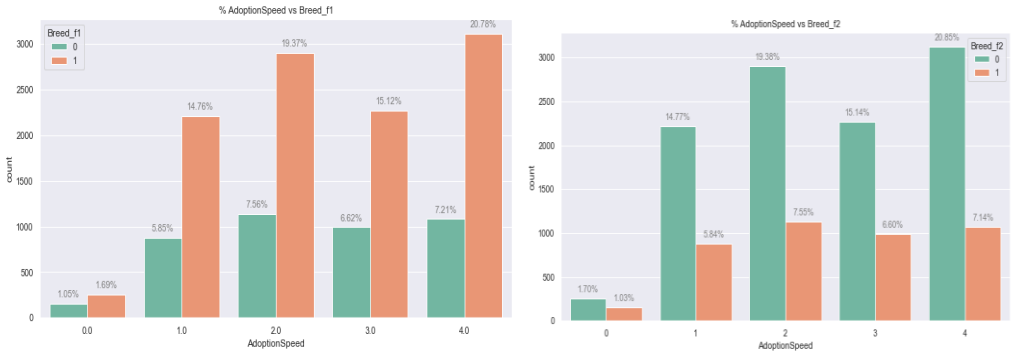


Fonte: autor

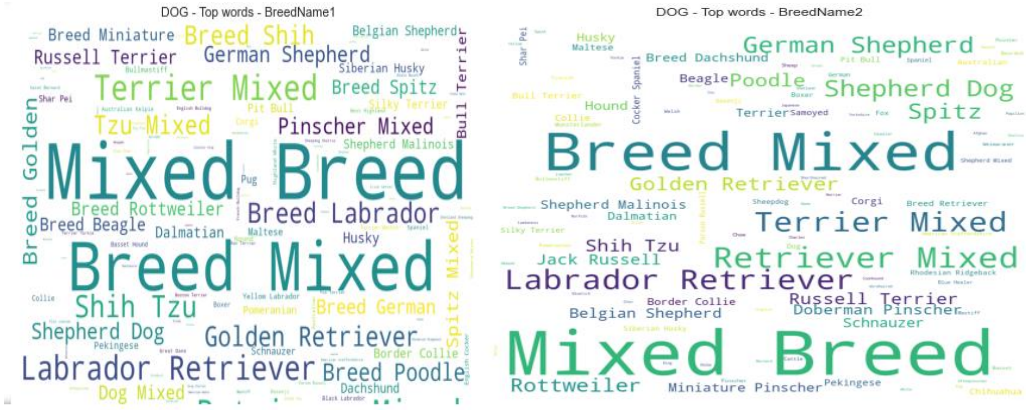


Fonte: autor

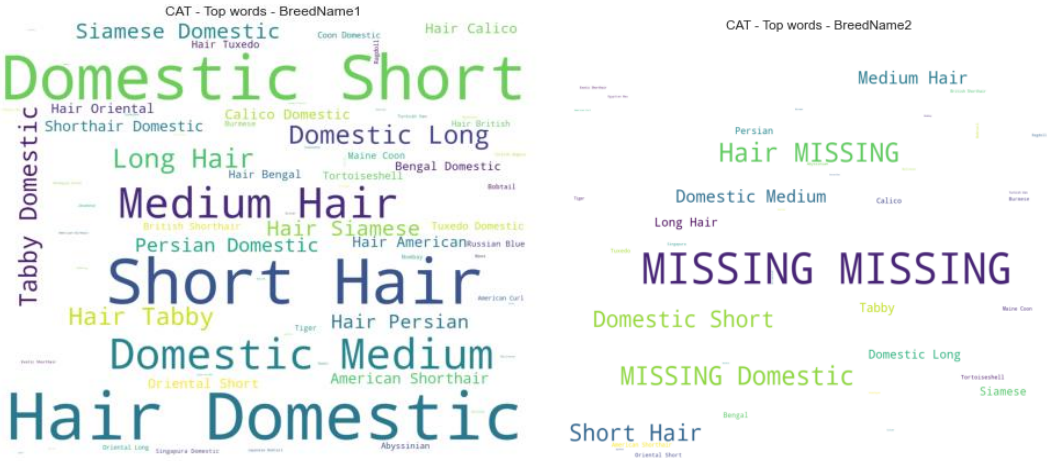
- **Breed1 e Bree2:** Lembrando que criamos variáveis na fase de tratamento de dados para juntar esses 2 atributos, onde **Breed_f1=1** significa que temos apenas informação no atributo Breed1 e não temos no Breed2; e **Breed_f2=1** é quando temos informação nos 2 atributos Breed1 e Breed2, senão, o valor fica 0. Então, a princípio temos mais informação da raça no atributo Breed1, mas será que válido a informação? Usando a função *WordCloud* e vimos que a informação em destaque é *Mixed* ou *Breed* para adoção de cachorros e *Domestic Hair, Short* para gatos;



Fonte: autor

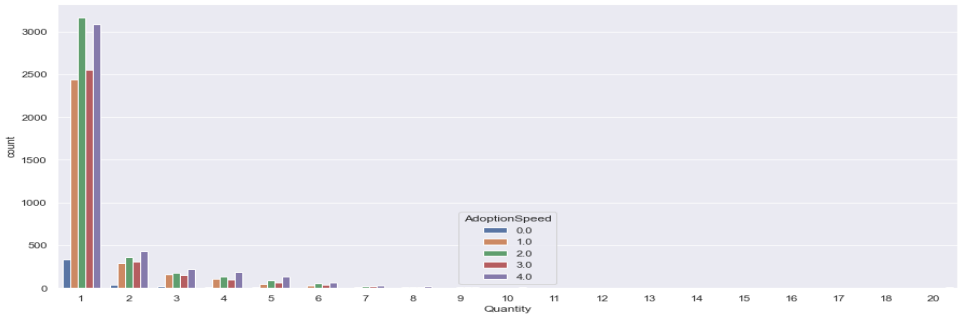


Fonte: autor

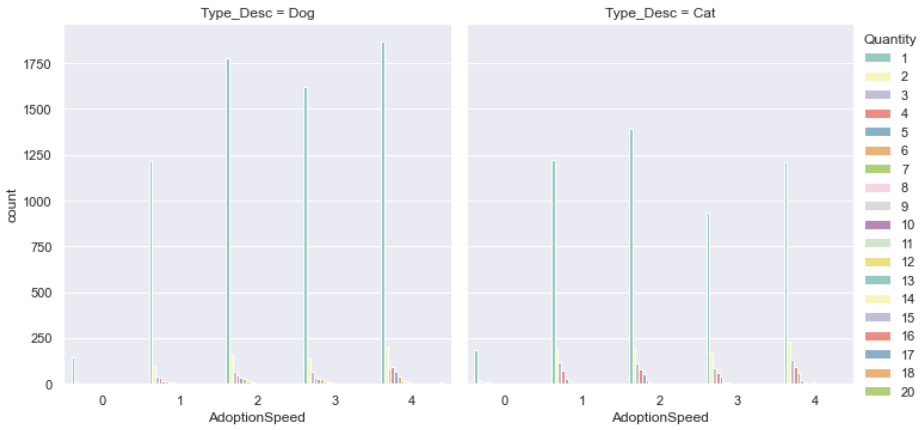


Fonte: autor

- **Quantity**: É o número de animais de estimação representado no perfil, ou seja, a maioria consta apenas 1 perfil na base de dados.



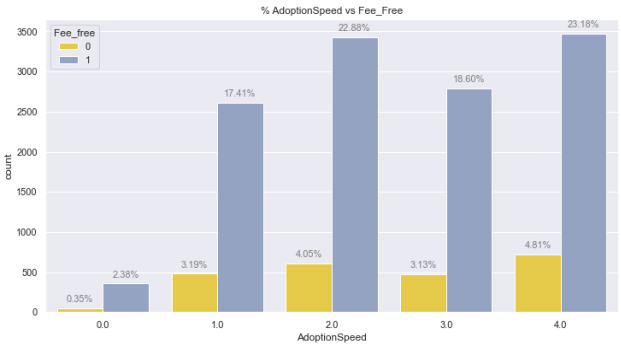
Fonte: autor



- **Fee**: Criamos um flag para identificar se tem taxa de adoção (0) ou se a adoção não tem taxa (1); e a maioria são adotados sem pagar taxa;

```
train[["Fee","Fee_free"]]
```

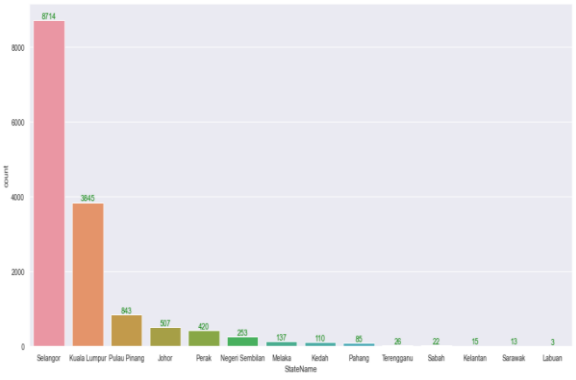
	Fee	Fee_free
0	100	0
1	0	1
2	0	1
3	0	1
4	300	0



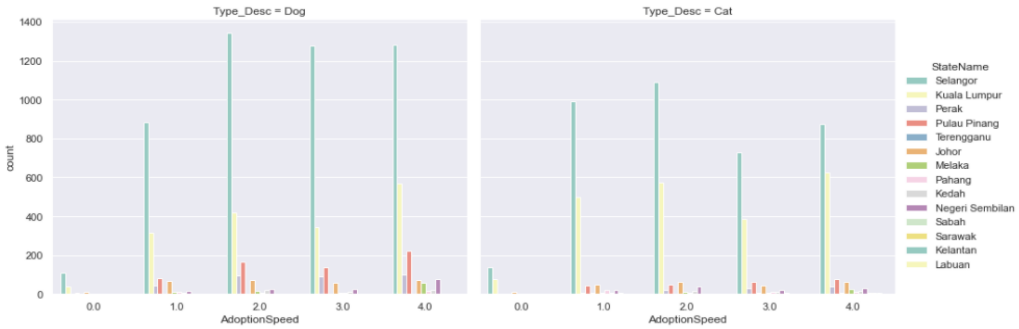
Fonte: autor

- **StateName**: Os estados mais populosos da Malásia como Selangor, Kuala Lumpur e Perak são os que mais adotam os animais.

	State	StateName	state_population
0	41326	Selangor	5.000.000
1	41326	Selangor	5.000.000
2	41326	Selangor	5.000.000
3	41326	Selangor	5.000.000
4	41326	Selangor	5.000.000
...	...	...	...
14988	41367	Kelantan	2.000.000
14989	41367	Kelantan	2.000.000
14990	41415	Labuan	99.500



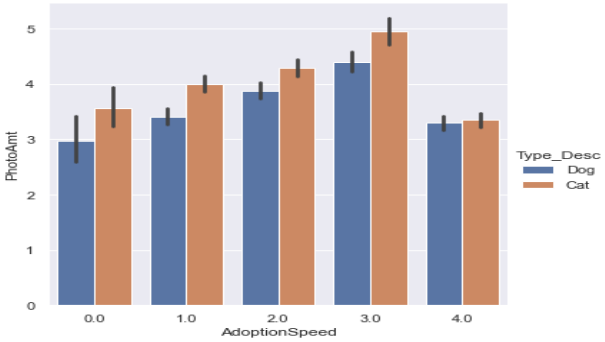
Fonte: autor



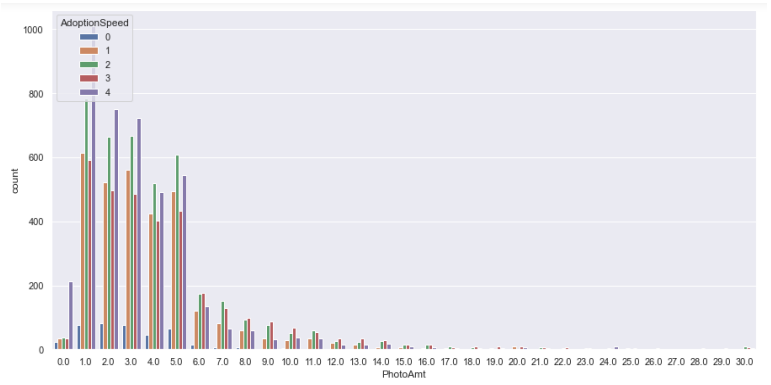
Fonte: autor

- **PhotoAmt**: Existem animais que não possuem fotos e outros que tem até 30 fotos, mas a média fica em 3 fotos; e os gatos são adotados mais facilmente com fotos.

```
: train['PhotoAmt'].describe()

: count    14993.000000
  mean         3.889215
  std          3.487810
  min          0.000000
  25%          2.000000
  50%          3.000000
  75%          5.000000
  max         30.000000
  Name: PhotoAmt, dtype: float64
```

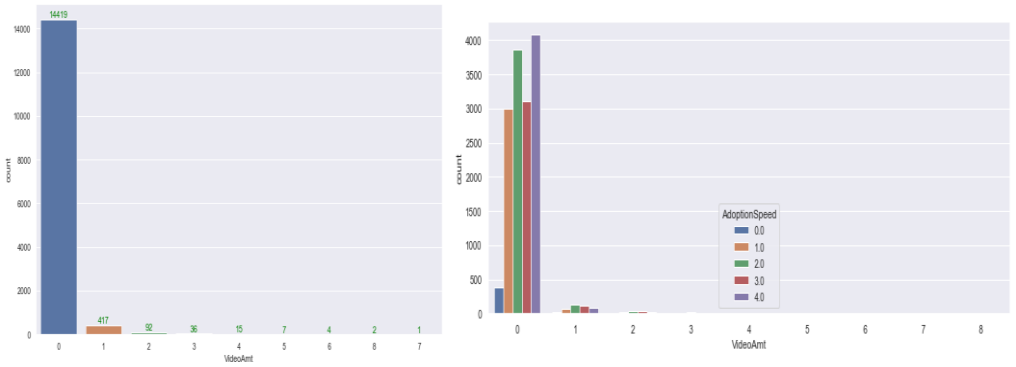


Fonte: autor



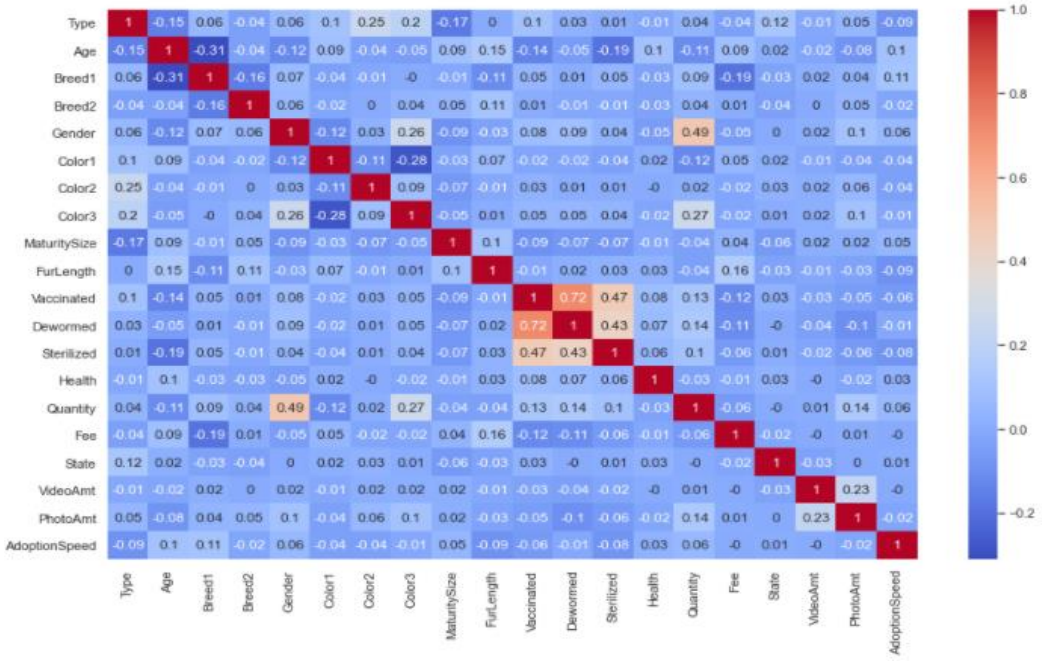
Fonte: autor

- **VideoAmt**: São poucos animais que possuem vídeos, mas isso aparentemente não interfere na velocidade da adoção;



Fonte: autor

E após fazermos a exploração e visualização dos dados, agora vamos verificar a relação entre os atributos e para isso utilizamos a ferramenta *Correlation Matrix* (matriz de correlação):



Fonte: autor

A matriz demonstra que não há nenhuma correlação significativa entre os atributos prevalecendo a tonalidade do azul (fraca). E a correlação com o *target AdoptionSpeed* os atributos *Age e Breed1* são os que se mostraram mais positivos, mas mesmo sendo uma das correlações mais altas da coluna, a relação ainda é fraca 0,1 e 011. As colunas *Dewormed, Vaccinated* e *Sterilized* apresentaram a melhor correlação entre si, em torno 0.3 a 0,7 (centro da matriz). E como esses atributos estão ligados a saúde então verificamos o atributo *Health* em relação *AdoptionSpeed* que também poderia ter uma boa correspondência, mas apresentou uma baixa correlação apenas 0,03.

Portanto, nessa análise dos dados podemos observar alguns pontos que podem afetar ou não a velocidade da adoção do animal de estimação:

- As fotos dos animais podem contribuir na rapidez da adoção e os gatos possuem mais fotos e são os mais adotados nos primeiros dias;
- Existem mais cachorros que gatos e os gatos são adotados mais rápidos nos primeiros dias/meses de vida;
- As fêmeas tanto para gatos ou cachorros são as preferidas na hora da adoção;
- Animais de estimação de porte médio são adotados mais rápido;

- Um animal com pelo curto tem mais chance de ser adotado;
- Raça tipo mista parece ser adotado mais rapidamente;
- As cores pretas, brancas e marrons são as mais escolhidas;
- A maioria são adotados sem ter que pagar taxa;
- A maioria dos gatos adotados não são vacinados; e os cachorros que não são vacinados é maior apenas nos primeiros dias de vida, logo depois a maioria são vacinados. Vacinados, desparasitados e esterilizados parecem não ter impacto na velocidade de adoção. E a maioria dos animais são saudáveis e não são castrados e desparasitado para adoção;
- Muitos não possuem vídeos, mas esse atributo parece não ter impacto na hora da adoção;
- A maioria dos animais de estimação são adotados mais rapidamente nos estados mais populoso da Malásia, exemplo, o estado de *Selangor* que tem uma população de 5 milhões .
- Animais de estimação mais velhos acabam sendo adotados mais lentamente;
- A idade parece não interferir na adoção, pois eles podem ser adotados no mesmo dia que são listados na plataforma independente de ser novo ou adulto;

5. Criação de Modelos de Machine Learning

Após análise e tratamento dos dados agora iremos aplicar alguns algoritmos de classificação para identificar através do atributo alvo quanto tempo demora para adoção dos animais de estimação. Atualmente, existem vários modelos e a ideia é aplicar em alguns desses modelos de aprendizado de máquina preditivas supervisionadas de classificação no conjunto de dados fornecidos.

A estratégia inicial foi executar os algoritmos sem os hiperparâmetros e, em seguida, ajustá-los conforme cada algoritmos os valores de hiperparâmetros. E depois de analisar essas duas execuções foi selecionado o modelo que obteve a melhor acurácia, e partir desse modelo, foi feito avaliação desse modelo e análise de melhoraria dos valores das métricas através de ajuste de valores de parâmetros,

avaliações do tipo: grau de importância dos recursos, curva ROC e da matriz de confusão.

Lembrando que estamos utilizando a ferramenta *Jupyter Notebook e* através da linguagem *Python* iremos fazer as classificações, mas antes de iniciarmos o aprendizado de máquina foi realizado a seleção dos atributos relevantes para melhor desempenho do modelo:

Fonte: autor

E repartimos aleatoriamente o *dataset* em conjunto de dados de treinamento (70%) e teste (30%) para treinarmos os modelos e também verificarmos as repartições na questão da porcentagem do *target Adoption Speed:*

Fonte: autor

E após a geração das amostras foram aplicados os modelos de aprendizado supervisionado utilizando a biblioteca de aprendizado de máquina *scikit-learn* com a visão do modelo com definição de valores de hiperparâmetros ou sem definição de valores, abaixo os valores das métricas gerado pelo *Classification Report* (relatório de classificação):

- KNeighborsClassifier: é um modelo simples e eficaz apresentando melhores valores para a categoria 4 e menor valor para categoria 0.

Sem valor hiperparâmetro

```
Acurária - treinamento de 53.53%
Acurária - previsão de 33.73%
Matriz de confusão:
 [[  5  42  39  16  21]
 [ 14 383 279 141 110]
 [ 23 341 413 223 211]
 [ 13 224 315 219 207]
 [ 17 242 325 178 497]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.07      0.04      0.05       123
           1       0.31      0.41      0.35       927
           2       0.30      0.34      0.32      1211
           3       0.28      0.22      0.25       978
           4       0.48      0.39      0.43      1259

    accuracy                           0.34      4498
   macro avg       0.29      0.28      0.28      4498
weighted avg       0.34      0.34      0.34      4498
```

Com valor hiperparâmetro

```
Acurária - treinamento de 98.57%
Acurária - previsão de 33.57%
Matriz de confusão:
 [[ 16  37  35  13  22]
 [ 39 308 274 167 139]
 [ 29 266 386 280 250]
 [ 20 172 261 283 242]
 [ 15 202 303 222 517]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.13      0.13      0.13       123
           1       0.31      0.33      0.32       927
           2       0.31      0.32      0.31      1211
           3       0.29      0.29      0.29       978
           4       0.44      0.41      0.43      1259

    accuracy                           0.34      4498
   macro avg       0.30      0.30      0.30      4498
weighted avg       0.34      0.34      0.34      4498
```

Fonte: autor

- GBMClassifier: os melhores valores das métricas estão para categoria 4.

Sem valor hiperparâmetro

```
Acurária - treinamento de 64.65%
Acurária - previsão de 39.24%
Matriz de confusão:
 [[  5  46  36   7  29]
 [  2 301 361  93 170]
 [  3 244 455 165 344]
 [  2 133 297 203 343]
 [  1 126 239  92 801]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.38      0.04      0.07       123
           1       0.35      0.32      0.34       927
           2       0.33      0.38      0.35      1211
           3       0.36      0.21      0.26       978
           4       0.47      0.64      0.54      1259

    accuracy                           0.39      4498
   macro avg       0.38      0.32      0.31      4498
weighted avg       0.38      0.39      0.38      4498
```

Com valor hiperparâmetro

```
Acurária - treinamento de 98.20%
Acurária - previsão de 37.33%
Matriz de confusão:
 [[  8  39  39  17  20]
 [  7 301 283 180 156]
 [  4 239 433 257 278]
 [  2 148 274 283 271]
 [  1 148 279 177 654]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.36      0.07      0.11       123
           1       0.34      0.32      0.33       927
           2       0.33      0.36      0.34      1211
           3       0.31      0.29      0.30       978
           4       0.47      0.52      0.50      1259

    accuracy                           0.37      4498
   macro avg       0.36      0.31      0.32      4498
weighted avg       0.37      0.37      0.37      4498
```

Fonte: autor

- DecisionTreeClassifier: a execução com ou sem definição do hiperparâmetro marcou a categoria 0 que ficou com o valor 0.

Sem valor hiperparâmetro

```
Acurária - treinamento de 98.64%
Acurária - previsão de 33.39%
Matriz de confusão:
 [[ 13  31  33  15  31]
 [ 29 288 250 179 181]
 [ 34 268 387 257 265]
 [ 18 160 272 294 234]
 [ 37 171 282 249 520]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.10      0.11      0.10       123
           1       0.31      0.31      0.31       927
           2       0.32      0.32      0.32      1211
           3       0.30      0.30      0.30       978
           4       0.42      0.41      0.42      1259

    accuracy                           0.33      4498
   macro avg       0.29      0.29      0.29      4498
weighted avg       0.34      0.33      0.33      4498
```

Com valor hiperparâmetro

```
Acurária - treinamento de 39.19%
Acurária - previsão de 37.62%
Matriz de confusão:
 [[  0  70  23   1  29]
 [  0 429 272  41 185]
 [  0 408 360  90 353]
 [  0 253 269  99 357]
 [  0 209 195  51 804]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       123
           1       0.31      0.46      0.37       927
           2       0.32      0.30      0.31      1211
           3       0.35      0.10      0.16       978
           4       0.47      0.64      0.54      1259

    accuracy                           0.38      4498
   macro avg       0.29      0.30      0.28      4498
weighted avg       0.36      0.38      0.35      4498
```

Fonte: autor

- GaussianNB: os valores das métricas aparentemente ficaram desiquilibradas quando foi incluído o hiperparâmetro, precisão e revocação com 0 ou 1 conforme a categoria.

Sem valor hiperparâmetro

```
Acurária - treinamento de 33.83%
Acurária - previsão de 34.28%
Matriz de confusão:
 [[  2  85  11  14  11]
 [ 24 523 196  71 113]
 [ 22 533 297 109 250]
 [ 22 342 215 184 215]
 [ 39 417 190  77 536]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.02      0.02      0.02       123
           1       0.28      0.56      0.37       927
           2       0.33      0.25      0.28      1211
           3       0.40      0.19      0.26       978
           4       0.48      0.43      0.45      1259

    accuracy                           0.34      4498
   macro avg       0.30      0.29      0.27      4498
weighted avg       0.37      0.34      0.33      4498
```

Com valor hiperparâmetro

```
Acurária - treinamento de 29.51%
Acurária - previsão de 28.77%
Matriz de confusão:
 [[   0    0   30    0   93]
 [   0    0  196    0  731]
 [   0    0  152    0 1059]
 [   0    0  174    1  803]
 [   0    0  118    0 1141]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       123
           1       0.00      0.00      0.00       927
           2       0.23      0.13      0.16      1211
           3       1.00      0.00      0.00       978
           4       0.30      0.91      0.45      1259

    accuracy                           0.29      4498
   macro avg       0.31      0.21      0.12      4498
weighted avg       0.36      0.29      0.17      4498
```

Fonte: autor

- RandomForestClassifier: a acurácia aumentou e apresentou os melhores valores das métricas paras as categorias, onde a categoria 4 continua sendo o que apresenta os melhores valores e categoria 0 a mais baixa.

Sem valor hiperparâmetro

```
Acurária - treinamento de 98.64%
Acurária - previsão de 39.26%
Matriz de confusão:
 [[  7  36  33  12  35]
 [  6 310 294 140 177]
 [  2 251 428 211 319]
 [  3 140 275 266 294]
 [  1 124 243 136 755]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.37      0.06      0.10       123
           1       0.36      0.33      0.35       927
           2       0.34      0.35      0.34      1211
           3       0.35      0.27      0.31       978
           4       0.48      0.60      0.53      1259

    accuracy                           0.39      4498
   macro avg       0.38      0.32      0.33      4498
weighted avg       0.38      0.39      0.38      4498
```

Com valor hiperparâmetro

```
Acurária - treinamento de 98.64%
Acurária - previsão de 39.42%
Matriz de confusão:
 [[  8  39  29  17  30]
 [  7 313 277 143 187]
 [  3 253 412 219 324]
 [  3 138 267 277 293]
 [  0 126 232 138 763]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.38      0.07      0.11       123
           1       0.36      0.34      0.35       927
           2       0.34      0.34      0.34      1211
           3       0.35      0.28      0.31       978
           4       0.48      0.61      0.53      1259

    accuracy                           0.39      4498
   macro avg       0.38      0.33      0.33      4498
weighted avg       0.39      0.39      0.38      4498
```

Fonte: autor

- ExtraTreesClassifier: apresentou os valores um pouco abaixo em comparação com *RandomForest*.

Sem valor hiperparâmetro

```
Acurária - treinamento de 98.64%
Acurária - previsão de 38.13%
Matriz de confusão:
 [[  7  37  34  15  30]
 [  8 303 278 146 192]
 [  8 253 431 216 303]
 [  4 147 287 278 262]
 [  2 163 249 149 696]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.24      0.06      0.09       123
           1       0.34      0.33      0.33       927
           2       0.34      0.36      0.35      1211
           3       0.35      0.28      0.31       978
           4       0.47      0.55      0.51      1259

    accuracy                           0.38      4498
   macro avg       0.35      0.32      0.32      4498
weighted avg       0.37      0.38      0.37      4498
```

Com valor hiperparâmetro

```
Acurária - treinamento de 98.64%
Acurária - previsão de 38.15%
Matriz de confusão:
 [[  7  34  34  16  32]
 [ 11 303 277 151 185]
 [  6 252 439 217 297]
 [  4 163 277 262 272]
 [  3 148 255 148 705]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.23      0.06      0.09       123
           1       0.34      0.33      0.33       927
           2       0.34      0.36      0.35      1211
           3       0.33      0.27      0.30       978
           4       0.47      0.56      0.51      1259

    accuracy                           0.38      4498
   macro avg       0.34      0.31      0.32      4498
weighted avg       0.37      0.38      0.37      4498
```

Fonte: autor

- LogisticRegression: apresentou o melhor valor de revocação para categoria 4, mas zerou as métricas para categoria 0.

Sem valor hiperparâmetro

```
Acurária - treinamento de 34.22%
Acurária - previsão de 33.86%
Matriz de confusão:
 [[  0  20  48   6  49]
 [  0  97 384  59 387]
 [  0  92 452  74 593]
 [  0  47 350 132 449]
 [  0  67 286  64 842]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       123
           1       0.30      0.10      0.16       927
           2       0.30      0.37      0.33      1211
           3       0.39      0.13      0.20       978
           4       0.36      0.67      0.47      1259

    accuracy                           0.34      4498
   macro avg       0.27      0.26      0.23      4498
weighted avg       0.33      0.34      0.30      4498
```

Com valor hiperparâmetro

```
Acurária - treinamento de 34.93%
Acurária - previsão de 34.44%
Matriz de confusão:
 [[  0  37  32   6  48]
 [  0 187 335  51 354]
 [  0 180 376  86 569]
 [  0  94 267 155 462]
 [  0 137 224  67 831]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       123
           1       0.29      0.20      0.24       927
           2       0.30      0.31      0.31      1211
           3       0.42      0.16      0.23       978
           4       0.37      0.66      0.47      1259

    accuracy                           0.34      4498
   macro avg       0.28      0.27      0.25      4498
weighted avg       0.34      0.34      0.31      4498
```

Fonte: autor

- XGBClassifier: os valores apresentados estão bem parecidos com o modelo *ExtraTreesClassifier*.

Sem valor hiperparâmetro

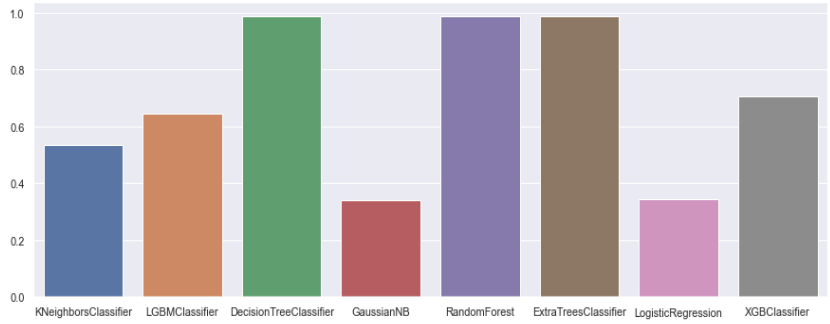
```
Acurária - treinamento de 70.46%
Acurária - previsão de 39.48%
Matriz de confusão:
 [[  6  39  41  10  27]
 [  2 309 345 102 169]
 [  2 241 463 191 314]
 [  2 136 303 227 310]
 [  0 125 258 105 771]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.50      0.05      0.09       123
           1       0.36      0.33      0.35       927
           2       0.33      0.38      0.35      1211
           3       0.36      0.23      0.28       978
           4       0.48      0.61      0.54      1259

    accuracy                           0.39      4498
   macro avg       0.41      0.32      0.32      4498
weighted avg       0.39      0.39      0.38      4498
```

Com valor hiperparâmetro

```
Acurária - treinamento de 98.64%
Acurária - previsão de 37.46%
Matriz de confusão:
 [[  9  35  38  17  24]
 [  4 322 287 150 164]
 [  4 244 408 261 294]
 [  6 140 280 272 280]
 [  2 150 269 164 674]]
Relatório de classificação:
              precision    recall  f1-score   support

           0       0.36      0.07      0.12       123
           1       0.36      0.35      0.35       927
           2       0.32      0.34      0.33      1211
           3       0.31      0.28      0.30       978
           4       0.47      0.54      0.50      1259

    accuracy                           0.37      4498
   macro avg       0.36      0.31      0.32      4498
weighted avg       0.37      0.37      0.37      4498
```

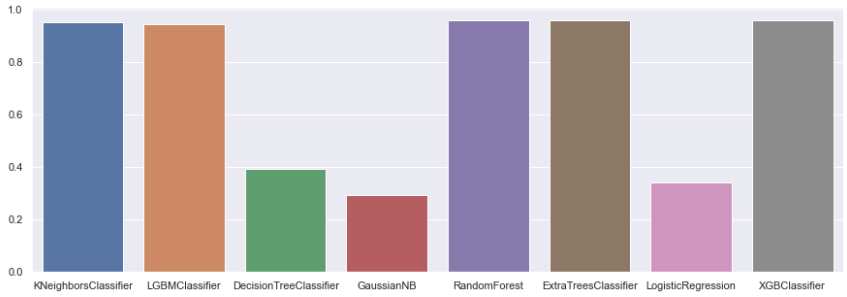
Fonte: autor

Então, gerando um gráfico de comparação para identificar qual desses métodos acima obteve o melhor desempenho sem ajuste do hiperparâmetro:



Fonte: autor

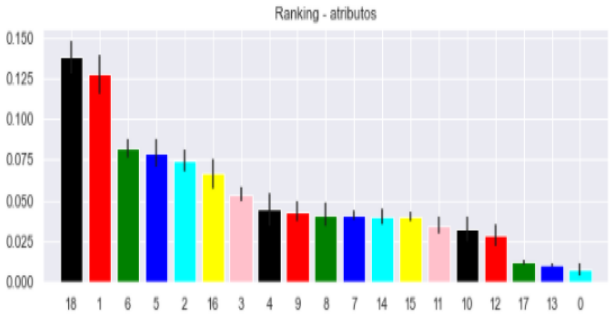
E outro gráfico com ajustes aleatórios dos hiperparâmetros de acordo com cada tipo de método. E como podemos verificar alguns métodos tiveram um melhor desempenho quando foram executados com valor do hiperparâmetro, como por exemplo o *KNeighborsClassifier*, porém, já para o caso do método *DecisionTreeClassifier* houve uma queda, detalhes no *Python*.



Fonte: autor

Dentre os métodos listados acima consideramos o *RandomForestClassifier* a princípio um bom método conforme os valores das métricas de acurácia, precisão e revocacão, então, iremos utilizá-lo como modelo de referência para tentar melhorar o seu desempenho. Sendo assim, o primeiro passo foi averiguar a importância dos recursos.

```
Ranking - atributos:
1. [18]: PhotoAmt (0.137768)
2. [1]: Age (0.127218)
3. [6]: Color2 (0.081997)
4. [5]: Color1 (0.079302)
5. [2]: Breed1 (0.074518)
6. [16]: State (0.066502)
7. [3]: Breed2 (0.053854)
8. [4]: Gender (0.044676)
9. [9]: FurLength (0.043242)
10. [8]: MaturitySize (0.041516)
11. [7]: Color3 (0.041185)
12. [14]: Quantity (0.040465)
13. [15]: Fee (0.040452)
14. [11]: Dewormed (0.034928)
15. [10]: Vaccinated (0.032622)
16. [12]: Sterilized (0.028718)
17. [17]: VideoAmt (0.012254)
18. [13]: Health (0.010512)
19. [0]: Type (0.008270)
```



Fonte: autor

Baseado no grau de importância dos recursos e pela análise dos dados que vimos nos passos anteriores, verificamos que as informações ligada a saúde (*Heath*) do animal não interfere na adoção, talvez seja porque todos estão saudáveis, então, removemos do *dataset (train)* essas informações que ficou denominado como *dataset* (*train2*) e com apenas 14 colunas que iremos trabalhar com ele para ver se melhora o desempenho do modelo.

```
#remover - train
train2 = train.drop(['FurLength','Vaccinated','Dewormed','Sterilized','Health','Type'], axis=1) #mantem valores
#train2 = train.drop(['VideoAmt','Type','Health'], axis=1) #cai os valores
#train2 = train.drop(['Color3','VideoAmt','Quantity','Vaccinated','Fee','Dewormed','Sterilized','Health'], axis=1) #cai
#train2 = train.drop(['Color3','VideoAmt','Quantity','Breed2','Fee','Gender','FurLength','MaturitySize','VideoAmt','Type'], axis=
train2
```

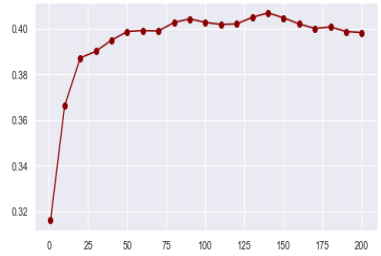
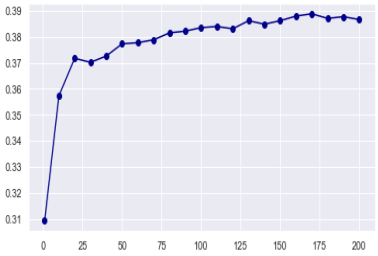
	Age	Breed1	Breed2	Gender	Color1	Color2	Color3	MaturitySize	Quantity	Fee	State	VideoAmt	PhotoAmt	AdoptionSpeed
0	3	167	0	0	0	6	0	0	0	29	2	0	1	2
1	1	136	0	0	0	1	0	1	0	0	12	0	2	0
2	1	175	0	0	1	6	0	1	0	0	2	0	7	3
3	4	175	0	1	0	1	0	1	0	36	12	0	8	2
4	1	175	0	0	0	0	0	1	0	0	2	0	3	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14988	2	137	0	2	0	0	0	1	3	0	2	0	3	2
14989	58	136	100	2	0	3	5	1	1	0	2	0	3	4
14990	2	136	102	2	4	5	5	2	4	11	2	0	5	3
14991	9	137	0	1	3	6	0	0	0	0	7	0	3	4
14992	1	175	134	0	1	0	0	1	0	0	5	0	1	3

14993 rows × 14 columns

Fonte: autor

Além disso, fizemos alguns testes para verificar qual o melhor valor de hiperparâmetro para o modelo *RandomForestClassifier* e os gráficos abaixos são referente a *n_estimators e random_state,* e graficamente o *n_estimators=140 e random_state=4 para o dataset* original *train* (vermelho) *e o dataset* ajustado *train2* (azul) ficou com o *n_estimators=170 e random_state=4.*

	Dataset – train2 (ajustado)

Dataset - train	
	

Fonte: autor

E executando novamente o método ou classificador *RandomForestClassifier* com o *dataset (train2)* ajustado, vimos que não houve melhoria dos valores da métricas acurácia, precisão e revocação em comparação com a execução com *dataset* original (*train*), segue o resultado do *Classification Report* (relatório de classificação).

Dataset - train	Dataset – train2 (ajustado)
Acurária - treinamento de 98.64%. Acurácia de 40.71%. Classification report:	Acurária - treinamento de 95.84% A acurácia de 38.88% Classification report:

	precision	recall	f1-score	support
0	0.400	0.065	0.112	123
1	0.372	0.337	0.353	927
2	0.354	0.378	0.366	1211
3	0.377	0.293	0.330	978
4	0.484	0.608	0.539	1259
accuracy			0.407	4498
macro avg	0.397	0.336	0.340	4498
weighted avg	0.400	0.407	0.397	4498

	precision	recall	f1-score	support
0	0.32	0.05	0.08	123
1	0.34	0.32	0.33	927
2	0.34	0.34	0.34	1211
3	0.35	0.29	0.31	978
4	0.47	0.59	0.53	1259
accuracy			0.39	4498
macro avg	0.36	0.32	0.32	4498
weighted avg	0.38	0.39	0.38	4498

Fonte: autor

Agora vamos interpretar as métricas que são importante para análise de desempenho do modelo no relatório de classificação *(Classficacion Report): accuracy, precision, recall e f1-score.* Vamos começar pela métrica **accuracy** (acurácia) que significa a proporção entre a observação prevista corretamente e o total de observações do modelo, nota-se que a maioria dos métodos ou classificadores até então executados nesse projeto estão com a acurácia entre a faixa de 30 à 40, sendo que, apenas o método *GaussianNB* ficou um pouco abaixo de 30.

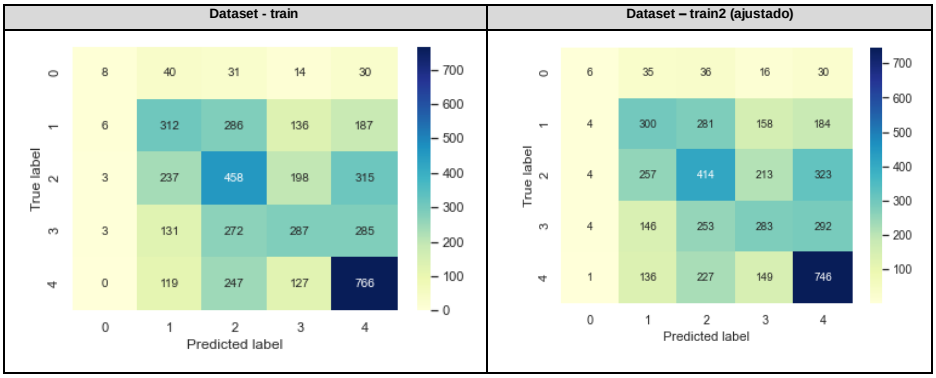
E em relação a métrica de ***precision*** (precisão) que é a proporção das observações positivas classificadas corretamente dentre as observações positivas que foram previstas, ou seja, a previsões corretas da categoria alvo são chamadas de **Verdadeiros Positivos – TP**(*true positive)*, e as

previsões incorretas para a categoria alvo são chamada de **Falsos Positivos – FP** (*false positive*), resumindo é a porcentagem dos dados que são relevantes. Vimos que alguns métodos como *GaussianNB ,ExtraTreesClassifier* e *LogisticRegression* a categoria=0 ficou baixa e para as outras categorias mantiveram na mesma faixa de valor para todos os métodos. E mesmo executando com o método até então avaliado o *RandomForestClassifier* com o *dataset* ajustado conforme o grau de importância do recurso, o valor da precisão para categoria=4 manteve-se inalterado e para a categoria=0 houve uma pequena queda.

Outra métrica importante é a **recall** (revocação) que é a proporção das observações positivas classificadas corretamente dentre as observações positivas verdadeiras, ou seja, são as previsões corretas da categoria **Verdadeiros Positivos – TP**(*true positive)* em relação a soma dos TP com os **Falsos Negativos – FN** (false negative) que é a categoria alvo que o modelo previu como outra categoria, resumindo é porcentagem do total de resultados relevantes classificado corretamente pelo algoritmo. Vimos que o método *GaussianNB* a categoria=4 ficou altissíma 0,91 e a categoria=0 ficou zerada com o ajuste do hiperparâmetro, e o mesmo aconteceu para o método *DecisionTreeClassifier* em que essas duas categorias ficaram com os valores destoados dos demais, a princípio vejo que seja uma questão *dataset* desbalanceado.

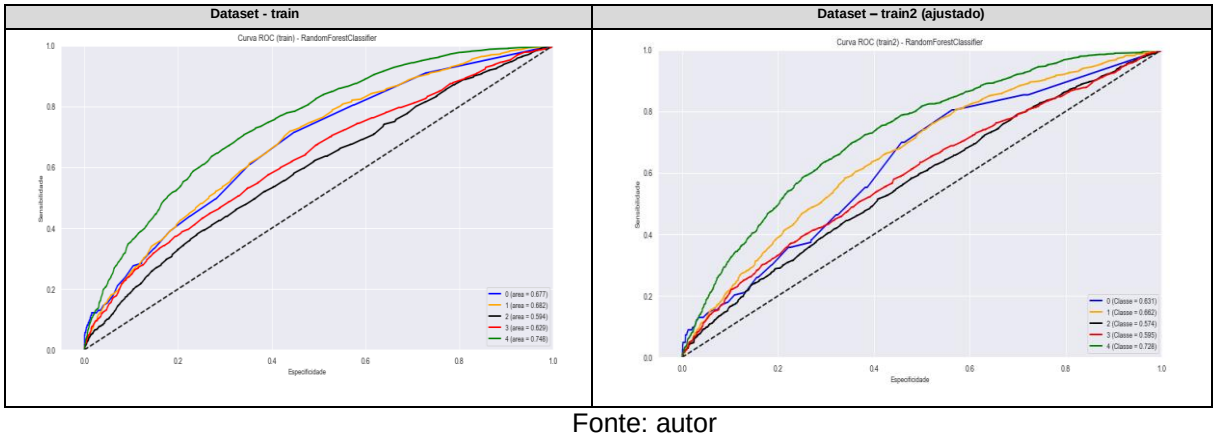
A métrica **F1-score** é uma média ponderada das métricas *precision* e *recall*, que nesse caso, o F1 é mais útil quando temos uma distribuição desigual entre as classes ou categorias.

Outra avaliação do comparativo dos *dataset's train* e *train2* foi a ***Confusion Matrix*** (matriz de confusão) onde temos em ambos os ***dataset's*** um aumento dos valores das categorias fora da diagonal, ou seja, o modelo classificou erroneamente. O primeiro valor da matriz 5x5 mostra quantas vezes o modelo previu corretamente categoria=0, valor igual 8 (*train*) e 6 (*train2*) e o último valor nos mostra o número de vezes que o modelo previu corretamente a categoria=4, valor igual 766 (*train*) e 746 (*train2*).



Fonte: autor

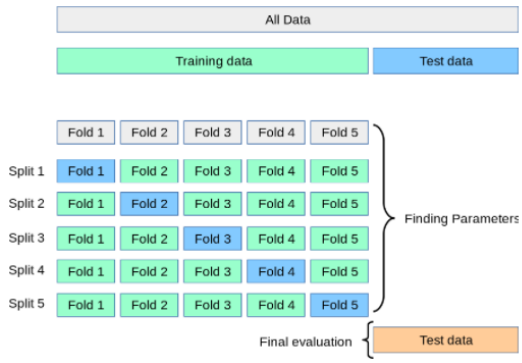
Existe também outra ferramenta chamada curva de ***ROC*** (*Receiver Operating Characteristic Curve*) que é uma outra forma de avaliar se o modelo é bom. E o modelo é considerado bom quando mais perto a linha de cada categoria alcançar o topo do eixo Y (1), que é a maior área sob curva ROC; e essa área é denominada como **AUC** (*Area Under the Curve*). O eixo y é a **Sensibilidadade** 100% (taxa de verdadeiros positivos) *versus* **Especificidade** 0% (taxa de falsos positivos), eixo X. A linha verde que é a categoria=4, indica o melhor resultado e os resultados abaixo da linha diagonal traceda preta são considerados classificadores ruins, e que nesse caso não houve nenhuma ocorrência.



Fonte: autor

Outra ferramenta que usamos foi a *Cross-Validation* (validação cruzada) que consiste em utilizar os dados de testes corretamente dentre o conjunto de treino.

Conforme a figura abaixo *cross-validation* em que os dados são divididos em 5 partes, o modelo usa o Fold (1) para teste e Fold(2,3,4 e 5) para treino, depois o modelo usa o Fold(2) para teste e Fold(1,3,4 e 5) para treino e assim sucessivamente.



Fonte: autor

E executando a validação cruzada para o *RandomForestClassifier* conforme os dataset's *train e train2*, e hiperparâmetros *n_estimators* (140 ou 170) e *random_state* (5 a 9) e a acurácia melhor classificada foi 40,36% e podendo ter o intervalo entre 37% a 41%.

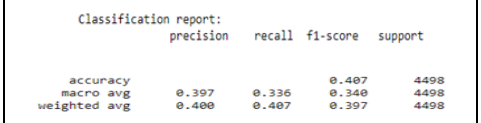
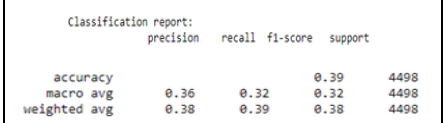
Dataset - train	Dataset – train2 (ajustado)
A acurácia média - random_state 5: 40.36% A acurária pode variar no intervalo - random_state 5: [38.92% ~ 41.80%] A acurácia média - random_state 6: 39.82% A acurária pode variar no intervalo - random_state 6: [37.32% ~ 42.31%] A acurácia média - random_state 7: 40.20% A acurária pode variar no intervalo - random_state 7: [38.28% ~ 42.12%] A acurácia média - random_state 8: 40.30% A acurária pode variar no intervalo - random_state 8: [38.56% ~ 42.05%] A acurácia média - random_state 9: 39.98% A acurária pode variar no intervalo - random_state 9: [38.59% ~ 41.38%]	A acurácia média - random_state 5: 37.80% A acurária pode variar no intervalo - random_state 5: [36.10% ~ 39.49%] A acurácia média - random_state 6: 37.68% A acurária pode variar no intervalo - random_state 6: [35.49% ~ 39.86%] A acurácia média - random_state 7: 37.07% A acurária pode variar no intervalo - random_state 7: [34.65% ~ 39.48%] A acurácia média - random_state 8: 37.44% A acurária pode variar no intervalo - random_state 8: [36.08% ~ 38.79%] A acurácia média - random_state 9: 37.62% A acurária pode variar no intervalo - random_state 9: [36.26% ~ 38.98%]

Fonte: autor

E a partir das métricas de **acurácia**, **precisão**, **revocação**, **curva ROC e validação cruzada** podemos concluir que dependendo do método treinado a questão do ajuste do hiperparâmetro pode influenciar nos valores das categorias, como por exemplo, *GaussianNB* com ajuste do hiperparâmetro e o *DecisionTreeClassifier* sem o ajuste do hiperparâmetro, as categorias foram erroneamente classificadas. E o *LogisticRegression* foi o único método que não sofreu grandes mudanças nos valores das métricas com ou sem ajuste do hiperparâmetro.

Bem, essas métricas acima são intuitivas de pontuar quando temos uma classificação binária, mas como nosso projeto temos uma classificação de multiclasses então temos que averiguar outras métricas que estão no *ClassificationReport* como:

- **Macro-averaging**: que é usado quando todas as classes/categorias precisarem ser tratadas igualmente para avaliar o desempenho geral do classificador em relação aos rótulos de classe/categoria mais frequentes. É a média aritmética da pontuação das classes individuais em relação à *precision, recall e f1-score*; todas as classes/categorias contribuem igualmente para a métrica média final;
- Weighted macro-averaging score: usado em caso de desequilíbrio de classe/categoria; e é calculado ponderando a pontuação de cada rótulo de classe/categoria pelo número de instâncias verdadeiras ao calcular a média;

Dataset - train	Dataset – train2 (ajustado)
	

Fonte: autor

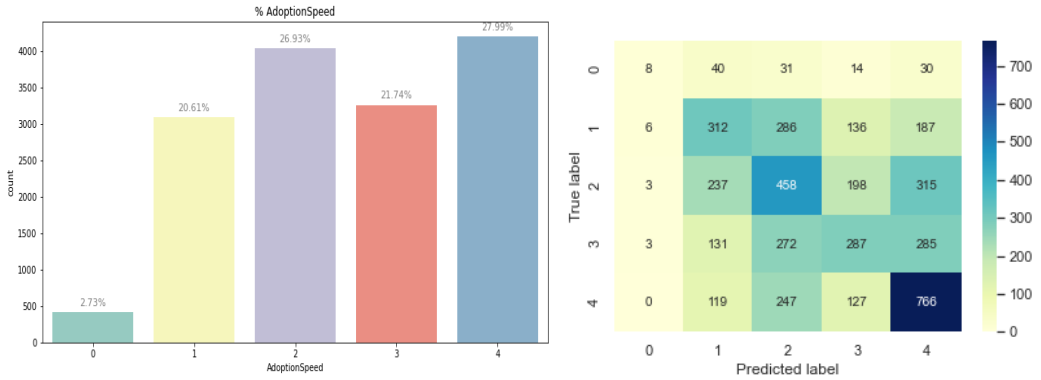
6. Apresentação dos Resultados

Conforme orientação da pós-graduação, primeiramente será apresentado o preenchimento do *workflow* motivador desse trabalho seguindo modelo de canvas proposto por *Vasandani* (clique aqui).

Título: Adoção do animal de estimação		
Problema	**Resultados / previsões**	**Aquisição de dados**
A velocidade para adotar um animal de estimação.	O foco é obter o modelo adequado em relação a velocidade de adoção do animal através do atributo alvo *AdoptionSpeed*. Previsão é incorporar dados de imagens e vídeos, além de aprimorar o ajuste do hiperparmetro e experimentar com outros modelos de aprendizados para aumentar o desempenho.	Os datasets são de Pets da plataforma Perfinder e estão disponiveis no *Kaggle*. A base corresponde ao ano de 2018.
Modelagem	**Avaliação de modelo**	**Preparação de dados**
Foram utilizados diferentes classificadores disponível na biblioteca sklearn utilizando a linguagem Python.	Existe um desbanlacemanto das classes e foi utilizado a ferramenta SMOTE para balancear o dataset. E também foram avaliados alguns ajustes de hiperparâmetro e grau de importância dos recursos , assim como , a utilização das ferramentas confusion matrix e cross valitadion.	Os dados foram tratados e analisados, não existem duplicação de linhas no dataset. Os atributos irrelevantes foram removidos e ajustado conforme grau de importância do recurso.

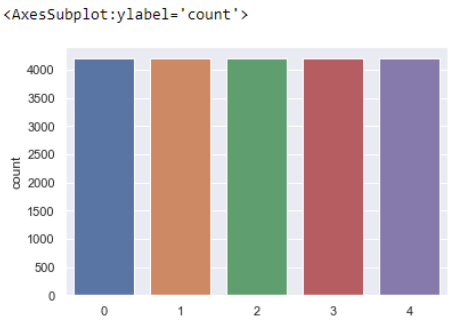
Fonte: autor

Conforme vimos anteriormente, o modelo *RandomForestClassifier* apresentou o melhor desempenho dentre todos os classificadores aprendizados de máquina treinado nesse projeto. Com o gráfico de distribuição da porcentagem do *target AdoptionSpeed* e a visualização da matriz de confusão que nos possibilita visualizar o desempenho do algoritmo de classificação, notamos que a categoria 4 e 2 foram as que tiveram mais acertos de previsões (valores da diagonal) e são justamente as categorias com maior porcentagem, e as outras categorias tiveram mais erros nas previsões (valores fora da diagonal da tabela), principalmente a categoria 0 que obteve apenas 8 acertos na previsão.



Fonte: autor

A categoria 0 (o animal de estimação foi adotado no mesmo dia em que foi listado) abrange apenas 3% do conjunto de dados, o que demonstra um desiquilíbrio com outras 4 categorias tornando a previsão para essa categoria mais difícil, e conforme as avaliações dos modelos essa categoria 0 apresentou os menores números. Então, utilizamos a ferramenta *SMOTE* para balancear o *dataset* (*train*) e executamos novamente o modelo e as ferramentas de análise de dados.



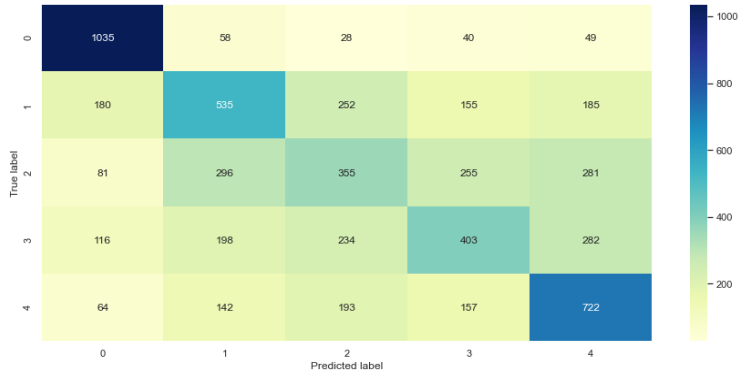
Fonte: autor

RandomForestClassifier: ao executar novamente o classificador a acurácia ficou com 49%, aumento de cerca de 10% em relação a execução anterior com o *dataset (train)* desbalanceado.

```
Acurária - treinamento de 98.62%.
Acurácia de 48.44%.
Classification report:
              precision    recall  f1-score   support

           0       0.70      0.86      0.77      1210
           1       0.44      0.41      0.42      1307
           2       0.33      0.28      0.30      1268
           3       0.40      0.33      0.36      1233
           4       0.48      0.56      0.52      1278

    accuracy                           0.48      6296
   macro avg       0.47      0.49      0.47      6296
weighted avg       0.47      0.48      0.47      6296
```

Fonte: autor

Matriz Confusão: os valores da diagonal aumentaram, ou seja, maior o número de acerto das previsões, e destacando agora a categoria 0 que acabou pontuando mais que a categoria 4, que até então era a categoria que sempre tinha a maior pontuação na tabela.



Fonte: autor

Grau de importância: em relação ao grau de importância dos recursos, a *PhotoAmt* e a *Age* permaneceram nos primeiros lugares do *ranking* e fazendo comparação com o *ranking* anterior notamos que a prioridade mudou para alguns recursos, mas a idade e a foto continuaram nas primeiras posições.

```
Ranking - atributos (balanced):
1. [1]: Age (0.125465)
2. [18]: PhotoAmt (0.122264)
3. [2]: Breed1 (0.097083)
4. [6]: Color2 (0.079204)
5. [5]: Color1 (0.076739)
6. [16]: State (0.070202)
7. [3]: Breed2 (0.057034)
8. [7]: Color3 (0.044041)
9. [9]: FurLength (0.041550)
10. [15]: Fee (0.041497)
11. [4]: Gender (0.039376)
12. [14]: Quantity (0.036876)
13. [11]: Dewormed (0.035737)
14. [8]: MaturitySize (0.035374)
15. [10]: Vaccinated (0.033967)
16. [12]: Sterilized (0.031637)
17. [0]: Type (0.014110)
18. [17]: VideoAmt (0.009716)
19. [13]: Health (0.008128)
```

versus

```
Ranking - atributos:
1. [18]: PhotoAmt (0.135018)
2. [1]: Age (0.128504)
3. [6]: Color2 (0.082656)
4. [5]: Color1 (0.080920)
5. [2]: Breed1 (0.074853)
6. [16]: State (0.067297)
7. [3]: Breed2 (0.053791)
8. [9]: FurLength (0.044101)
9. [4]: Gender (0.043161)
10. [7]: Color3 (0.041517)
11. [8]: MaturitySize (0.041144)
12. [15]: Fee (0.040263)
13. [14]: Quantity (0.040103)
14. [11]: Dewormed (0.033963)
15. [10]: Vaccinated (0.031540)
16. [12]: Sterilized (0.030354)
17. [17]: VideoAmt (0.012254)
18. [13]: Health (0.010307)
19. [0]: Type (0.008253)
```

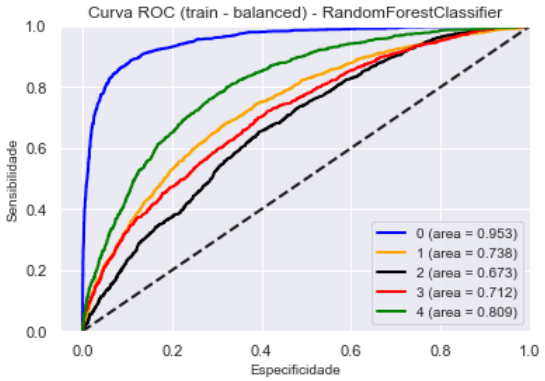
Fonte: autor

Validação Cruzada: houve um aumento da acurácia ficando agora em 48,85% e também podendo ficar no intervalo de 47% a 50% conforme o valor do hiperparâmetro *random_state*.

```
A acurácia média - random_state 5: 48.78%
A acurária pode variar no intervalo - random_state 5: [48.14% ~ 49.41%]

A acurácia média - random_state 6: 48.47%
A acurária pode variar no intervalo - random_state 6: [47.39% ~ 49.55%]

A acurácia média - random_state 7: 48.85%
A acurária pode variar no intervalo - random_state 7: [47.84% ~ 49.87%]

A acurácia média - random_state 8: 48.68%
A acurária pode variar no intervalo - random_state 8: [48.10% ~ 49.26%]

A acurácia média - random_state 9: 48.49%
A acurária pode variar no intervalo - random_state 9: [48.23% ~ 48.76%]
```

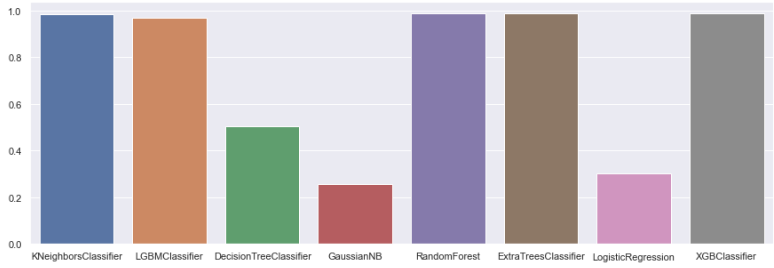
Fonte: autor

Curva ROC: a categoria 0 que estava mais baixa em todos as classificações, agora ela apresenta o melhor alcance do topo (y), e a categoria 4 manteve-se na segunda posição no alcance do topo.



Fonte: autor

Modelos/Classificarodres: o gráfico dos classificadores com o dataset (*train2*) balanceado esta bem parecido com o gráfico anterior do dataset com ajuste de hiperparâmetro (*train*), mas a diferença foi que agora com o *dataset* balanceado a acurácia para a maioria dos modelos ficou um pouco mais alto, entre 40% a 50%.



Fonte: autor

Infelizmente não existe uma fórmula automática de saber qual é o melhor modelo para um determinado *dataset*, a proposta é fazer ajustes e avaliações dos classificadores para adquirir conhecimento das informações para a melhor escolho do modelo. E utilizamos nesse projeto algumas técnicas e ferramentas como: eliminação de recursos/colunas do *dataset* para ajudar na regularização do modelo, observando que ajustar excessivamente também pode tornar um modelo ruim e avaliação das métricas geradas no *Classification Report* (relatório de classificação), matriz de confusão e outros.

Nosso objetivo era desenvolver algoritmos para prever a melhor velocidade de adoção dos animais de estimação e qual seria o perfil do animal de estimação que era mais adotado, além de, adquirir e melhorar a compreensão dos dados. E com isso, vimos que a base de dados estava desbalanceada para categoria 0 (*adoptionspeed*=0) e que após o balanceamento dos dados das categorias, o resultado do modelo acabou melhorando, ou seja, a velocidade de adoção do animal ficou em menos de 1 ou em 1 dia, *adoptionspeed*=0. E tendo em vista o comportamento dos dados em relação às ferramentas de avaliação utilizadas até o momento, os atributos de *PhotoAmt* e *Age* são os fatores do perfil do animal que ajudam a melhorar o desempenho do modelo. Um exemplo disso, são os gatos que tem mais *PhotoAmt* que os cachorros e são os mais rápidos a serem adotados, *adoptionspeed*=0 (ficam apenas 1 dia no abrigo). Sendo assim, nesse projeto vimos que um *dataset* balanceado, tratado e equilibrado na quantidade de atributos/colunas, tal como, ajuste corretos dos hiperparâmetros e utilização das ferramentas adequadas de validação podem melhorar o desempenho e na escolha do modelo. E no futuro próximo gostaríamos de experimentar novos modelos de aprendizado de máquina para tentar aumentar o desempenho; ter a possibilidade de treinar o modelo com uma nova base de dados para verificar a inexistência de *overfitting* que é um problema quando o modelo não entende novos dados, pois só entende os dados treinados; e incorporar dados de imagens dos animais no nosso aprendizado de máquina, assim como, trabalhar mais com os ajustes de hiperparâmetros.

7. Links

Vídeo de apresentação do projeto:
https://www.youtube.com/watch?v=mRLKwFEjaeE

Script do projeto:
https://github.com/sumirebsb/Projeto_PUC_CienciaDeDados_BigData/

Repositório do projeto:
https://www.kaggle.com/c/petfinder-adoption-prediction/data

REFERÊNCIAS

El rol de los perros y gatos durante la pandemia. **Affinity-Fundacion**. https://www.fundacion-affinity.org/observatorio/el-rol-de-los-perros-y-gatos-durante-la-pandemia. Acesso abril de 2021.

Professora da UFU pesquisa a relação entre tutores e seus pets na pandemia da Covid-19. **Ufu**,2020. http://www.comunica.ufu.br/noticia/2020/06/professora-da-ufu-pesquisa-relacao-entre-tutores-e-seus-pets-na-pandemia-da-covid-19 Acesso abril de 2021. Acesso abril de 2021.

Animais de estimação podem ajudar na saúde mental ao reduzir a solidão no isolamento. **Forbes**, 2020. https://forbes.com.br/forbessaude/2020/09/animais-de-estimacao-podem-ajudar-na-saude-mental-ao-reduzir-a-solidao-no-isolamento/. Acesso abril de 2021.

Ter animais de estimação pode aliviar estresse do isolamento na pandemia. **Galileu**, 2020.https://revistagalileu.globo.com/Sociedade/Comportamento/noticia/2020/09/ter-animais-de-estimacao-pode-aliviar-estresse-do-isolamento-na-pandemia.html. Acesso abril de 2021.

Plataforma de adoção de animais de estimação. **PetFinder.my**, 2008. Disponível em: https://www.petfinder.my/. Acesso abril de 2021.

Avaliar os resultados do experimento de machine learning automatizado. https://docs.microsoft.com/pt-br/azure/machine-learning/how-to-understand-automated-ml Acesso abril de 2021.

Dúvidas e Erros. https://stackoverflow.com/questions/ .Acesso abril de 2021.

How not to use random forest. https://medium.com/@toma_74920 Acesso abril de 2021.

Scikit-learn Machine Learning in Python https://scikit-learn.org/stable/ Acesso abril de 2021.

https://machinelearningmastery.com/ Acesso abril de 2021.

https://www.python.org/. Acesso abril de 2021.

https://matplotlib.org/. Acesso abril de 2021.

https://seaborn.pydata.org/. Acesso abril de 2021.