EDA And Baseline on Motor Temperature Estimation

In [1]:

```
!pip install seaborn
!pip install scikit-learn==0.23.2
```

```
Requirement already satisfied: seaborn in c:\users\pedro\anaconda3\lib\site-
packages (0.11.0)
Requirement already satisfied: pandas>=0.23 in c:\users\pedro\anaconda3\lib
\site-packages (from seaborn) (1.1.3)
Requirement already satisfied: numpy>=1.15 in c:\users\pedro\anaconda3\lib\s
ite-packages (from seaborn) (1.19.2)
Requirement already satisfied: matplotlib>=2.2 in c:\users\pedro\anaconda3\l
ib\site-packages (from seaborn) (3.3.2)
Requirement already satisfied: scipy>=1.0 in c:\users\pedro\anaconda3\lib\si
te-packages (from seaborn) (1.5.2)
Requirement already satisfied: pytz>=2017.2 in c:\users\pedro\anaconda3\lib
\site-packages (from pandas>=0.23->seaborn) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\pedro\anac
onda3\lib\site-packages (from pandas>=0.23->seaborn) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
c:\users\pedro\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(2.4.7)
Requirement already satisfied: certifi>=2020.06.20 in c:\users\pedro\anacond
a3\lib\site-packages (from matplotlib>=2.2->seaborn) (2020.6.20)
Requirement already satisfied: cycler>=0.10 in c:\users\pedro\anaconda3\lib
\site-packages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\pedro\anaconda3\lib
\site-packages (from matplotlib>=2.2->seaborn) (8.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\pedro\anaconda3
\lib\site-packages (from matplotlib>=2.2->seaborn) (1.3.0)
Requirement already satisfied: six>=1.5 in c:\users\pedro\anaconda3\lib\site
-packages (from python-dateutil>=2.7.3->pandas>=0.23->seaborn) (1.15.0)
Requirement already satisfied: scikit-learn==0.23.2 in c:\users\pedro\anacon
da3\lib\site-packages (0.23.2)
Requirement already satisfied: joblib>=0.11 in c:\users\pedro\anaconda3\lib
\site-packages (from scikit-learn==0.23.2) (0.17.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\pedro\anacon
da3\lib\site-packages (from scikit-learn==0.23.2) (2.1.0)
Requirement already satisfied: scipy>=0.19.1 in c:\users\pedro\anaconda3\lib
\site-packages (from scikit-learn==0.23.2) (1.5.2)
Requirement already satisfied: numpy>=1.13.3 in c:\users\pedro\anaconda3\lib
\site-packages (from scikit-learn==0.23.2) (1.19.2)
```

Importando Bibliotecas

In [2]:

```
import numpy as np #Biblioteca "matemática"
import pandas as pd #Biblioteca para manipulação e análise de dados
import matplotlib.pyplot as plt #Extenção da biblioteca que faz a pltagem de gráficos e pon
from matplotlib.colors import rgb2hex
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import neighbors
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
%matplotlib inline
import os #Funcionalidade simplificadas de sistema operacionais
from sklearn.model_selection import cross_val_score # Cross Validation Function.
from sklearn.model_selection import KFold # KFold Class.
from sklearn.linear_model import LinearRegression # Linear Regression class.
from sklearn.metrics import mean_squared_error
import sklearn.metrics

print(os.listdir())
plt.style.use('bmh')
```

```
['.ipynb_checkpoints', 'activate.csh', 'activate.fish', 'activate.nu', 'acti
vate.ps1', 'activate.txt', 'activate_this.py', 'alembic.txt', 'archive.zip',
'automl_inicio.ipynb', 'AutoSklearn - Eletric Motor.ipynb', 'Book - Hands-on
atuomated machine learning.ipynb', 'Curso Data science.ipynb', 'cygdb.txt',
'cython.txt', 'cythonize.txt', 'dask-scheduler.txt', 'dask-ssh.txt', 'dask-w
orker.txt', 'databricks.txt', 'datahr.csv', 'dbfs.txt', 'deactivate.nu', 'di
stro.txt', 'Eletric Motor_Autosklearn-Copy1.ipynb', 'Eletric Motor_Autosklea
rn-Copy2.ipynb', 'Eletric Motor_Autosklearn.ipynb', 'Eletric Motor_Autosklea
rn_H2O.ipynb', 'Eletric Motor_pycaret.ipynb', 'f2py.txt', 'f2py3.8', 'f2py3.
txt', 'find_similar_images.py', 'flask.txt', 'fonttools.txt', 'futurize.tx
t', 'gunicorn.txt', 'htmlmin.txt', 'ipython.txt', 'ipython3.txt', 'jsonschem
a.txt', 'jupyter-bundlerextension.txt', 'jupyter-console.txt', 'jupyter-deja
vu.txt', 'jupyter-execute.txt', 'jupyter-kernel.txt', 'jupyter-kernelspec.tx
t', 'jupyter-migrate.txt', 'jupyter-nbconvert.txt', 'jupyter-nbextension.tx
t', 'jupyter-notebook.txt', 'jupyter-qtconsole.txt', 'jupyter-run.txt', 'jup
yter-serverextension.txt', 'jupyter-troubleshoot.txt', 'jupyter-trust.txt',
'jupyter.txt', 'logs.log', 'mako-render.txt', 'measures_v2.csv', 'melb_data.
csv', 'mlflow.txt', 'mlruns', 'nltk.txt', 'normalizer.txt', 'numba.txt', 'pa
ndas_profiling.txt', 'pasteurize.txt', 'phik_trial.txt', 'pip-3.8', 'pip.tx
t', 'pip3.8', 'pip3.txt', 'plac_runner.py', 'pycc.txt', 'pyftmerge.txt', 'py
ftsubset.txt', 'pygmentize.txt', 'python', 'python3', 'python3.8', 'send2tra
sh.txt', 'smac.txt', 'spacy', 'sqlformat.txt', 'tabulate.txt', 'tqdm.txt',
'ttx.txt', 'wheel-3.8', 'wheel.txt', 'wheel3.8', 'wheel3.txt', 'wordcloud_cl
i.txt', 'wsdump.txt', '__pycache__']
```

Lendo o arquivo

In [3]:

```
df=pd.read_csv('measures_v2.csv', usecols=[0,1,2,3,4,5,6,7,8,9,10,11])
target = df.pop('pm') #Temperatura do rotor
df = pd.concat([df, target], axis=1)
df = df.sample(frac=1,random_state=0) #embaralha os dados do dataframe #Ajuda a previnir o
df.reset_index(drop=True, inplace=True) #Faz com que o Index volte a ser o que era antes
```

In [4]:

```
df.head()
```

Out[4]:

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	
0	41.938923	18.744030	66.684830	-123.478027	46.080647	4749.964355	-187.964111	7
1	-0.431508	59.902590	85.079312	-0.878644	76.299257	0.057160	-2.000745	
2	-1.541598	33.149664	48.669293	-0.333442	45.330586	0.001482	-2.000673	
3	42.387482	44.949261	104.791174	-123.337533	90.274398	5112.368164	-181.587703	6
4	15.335679	18.755226	113.366333	-130.067474	84.144737	3999.963135	-205.157623	9

In [5]:

```
split_index=int(len(df) * 0.75)

train_df = df[:split_index] #Primeiros 75%
test_df = df[split_index:] #outros 25% restantes

train_df.info()
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 998112 entries, 0 to 998111
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   u_q             998112 non-null  float64
 1   coolant         998112 non-null  float64
 2   stator_winding  998112 non-null  float64
 3   u_d             998112 non-null  float64
 4   stator_tooth    998112 non-null  float64
 5   motor_speed     998112 non-null  float64
 6   i_d             998112 non-null  float64
 7   i_q             998112 non-null  float64
 8   stator_yoke     998112 non-null  float64
 9   ambient         998112 non-null  float64
 10  torque          998112 non-null  float64
 11  pm              998112 non-null  float64
dtypes: float64(12)
memory usage: 91.4 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 332704 entries, 998112 to 1330815
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   u_q             332704 non-null  float64
 1   coolant         332704 non-null  float64
 2   stator_winding  332704 non-null  float64
 3   u_d             332704 non-null  float64
 4   stator_tooth    332704 non-null  float64
 5   motor_speed     332704 non-null  float64
 6   i_d             332704 non-null  float64
 7   i_q             332704 non-null  float64
 8   stator_yoke     332704 non-null  float64
 9   ambient         332704 non-null  float64
 10  torque          332704 non-null  float64
 11  pm              332704 non-null  float64
dtypes: float64(12)
memory usage: 30.5 MB
```

Retira a última coluna que é no target do modelo de treinamento e modelos de teste

In [6]:

```
X_train = train_df.to_numpy()[:, :-1]
y_train = train_df.to_numpy()[:, -1]

X_test = test_df.to_numpy()[:, :-1]
y_test = test_df.to_numpy()[:, -1]
```

Criando o modelo para o treinamento do algorítmo

In [8]:

```
knn_model= neighbors.KNeighborsRegressor(n_neighbors=2, p=1, weights='distance')

knn_treino=knn_model.fit(X_train,y_train)
knn_teste=knn_model.fit(X_train,y_train)
```

Fazendo as predições dos valores

In [9]:

```
Pred_train_y =knn_model.predict(X_train)
```

In [10]:

```
Pred_test_y =knn_model.predict(X_test)
```

Observando o resultado das predições a partir do R^2 e o Mean Squared Error

In [11]:

```

print("Scores R2 de treino", sklearn.metrics.r2_score(y_train,Pred_train_y))
print("Scores R2 de teste", sklearn.metrics.r2_score(y_test,Pred_test_y))
```

```
Scores R2 de treino 1.0
Scores R2 de teste 0.9716096096469911
```

In [12]:

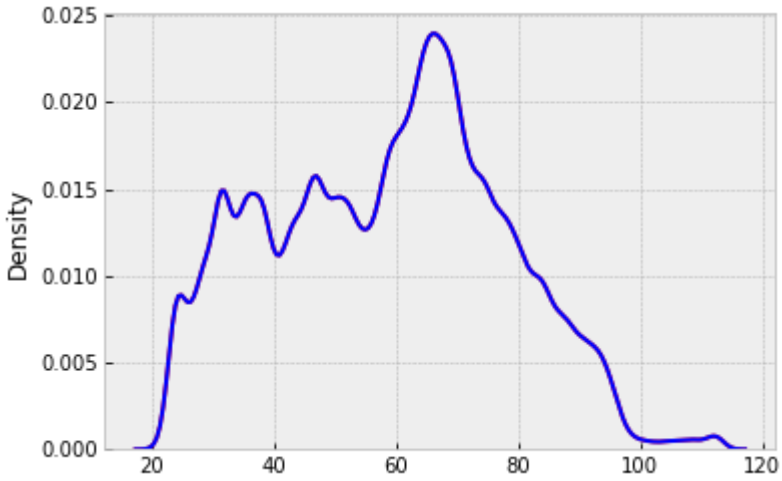
```
MSE_treino=sklearn.metrics.mean_squared_error(y_train, Pred_train_y)

MSE_teste=sklearn.metrics.mean_squared_error(y_test, Pred_test_y)

print("Erro quadrático Médio Treino", MSE_treino)

print("Erro quadrático Médio Teste", MSE_teste)
```

```
Erro quadrático Médio Treino 0.0
Erro quadrático Médio Teste 10.248833607996966
```

In [13]:

```
#Dados de treino
ax1 = sns.distplot(y_train, hist=False, color="r", label="Valor real")
sns.distplot(Pred_train_y, hist=False, color="b", label="Valor do treino" , ax=ax1);
```

```
C:\Users\pedro\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `kdeplot` (an axes-level function for
kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\pedro\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `kdeplot` (an axes-level function for
kernel density plots).
  warnings.warn(msg, FutureWarning)
```



Dados de Teste

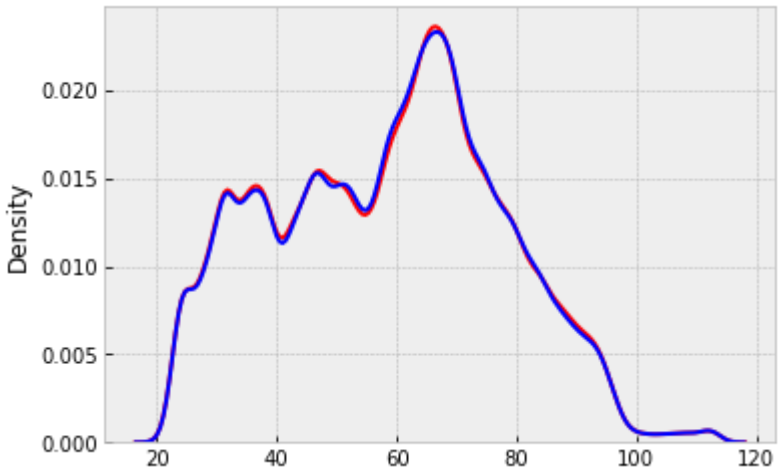
In [14]:

```
ax1 = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
sns.distplot(Pred_test_y, hist=False, color="b", label="Fitted Values" , ax=ax1)
```

```
C:\Users\pedro\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `kdeplot` (an axes-level function for
kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\pedro\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `kdeplot` (an axes-level function for
kernel density plots).
  warnings.warn(msg, FutureWarning)
```

Out[14]:

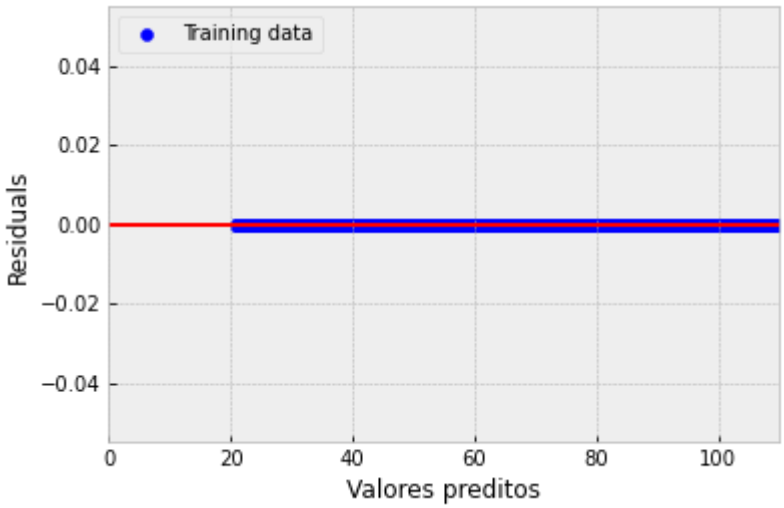
```
<AxesSubplot:ylabel='Density'>
```



Dados do treino

In [29]:

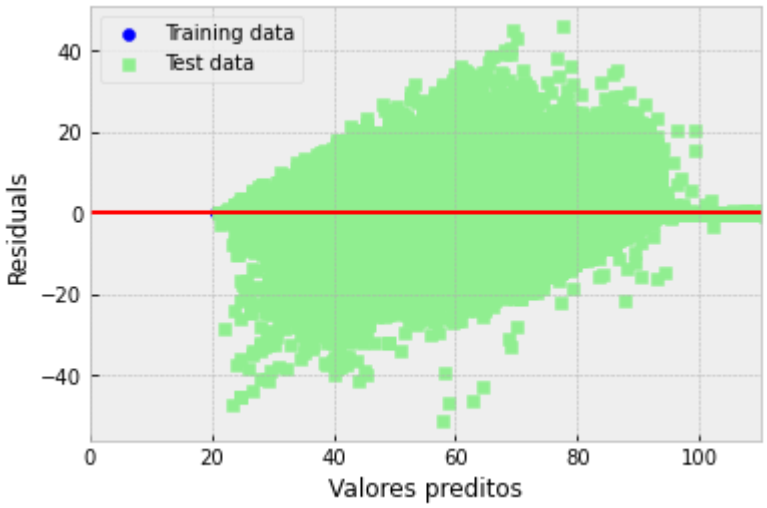
```
plt.scatter(Pred_train_y,Pred_train_y-  y_train,c='blue', marker='o', label='Training data'
plt.xlabel('Valores preditos')
plt.ylabel('Residuals')
plt.legend(loc='upper left')
plt.hlines(y=0, xmin=0, xmax=110, lw=2, color='red')
plt.xlim([0, 110])
plt.show()
```



Dados do treino e teste

In [32]:

```
plt.scatter(Pred_train_y,Pred_train_y-  y_train,c='blue', marker='o', label='Training data'
plt.scatter(Pred_test_y, Pred_test_y - y_test,c='lightgreen', marker='s', label='Test data'
plt.xlabel('Valores preditos')
plt.ylabel('Residuals')
plt.legend(loc='upper left')
plt.hlines(y=0, xmin=0, xmax=110, lw=2, color='red')
plt.xlim([0, 110])
plt.show()
```



Validação Cruzada

In [17]:

```
df=pd.read_csv('measures_v2.csv', usecols=[0,1,2,3,4,5,6,7,8,9,10,11])
target = df.pop('pm') #Temperatura do rotor
df = pd.concat([df, target], axis=1)
X_train = train_df.to_numpy()[:, :-1]
y_train = train_df.to_numpy()[:, -1]
X = test_df.to_numpy()[:, :-1]
y = test_df.to_numpy()[:, -1]
```

In [27]:

```
knn_model= neighbors.KNeighborsRegressor(n_neighbors=2, p=1, weights='distance')
kfold  = KFold(n_splits=10, shuffle=True) # shuffle=True, Shuffle (embaralhar) the data.
result = cross_val_score(knn_model, X, y, cv = kfold)

print("K-Fold (R^2) Scores: {0}".format(result))
print("Média do R^2 para a validação cruzada K-Fold: {0}".format(result.mean()))
```

```
K-Fold (R^2) Scores: [0.95689812 0.95807746 0.95758492 0.95751208 0.95800317
0.95902617
 0.95655186 0.95715746 0.95728968 0.95755505]
Média do R^2 para a validação cruzada K-Fold: 0.9575655962893705
```