Salut là-bas!

Или же просто привет всем) Сегодня я решил самостоятельно заняться каким-нибудь не очень большим исследованием данных, которые почти никто никогда не рассматривал вообще.
Освновная тема, вообще, которую мы сегодня вообще должны рассмотреть, и особенно просто понять, Как использовать ***Разбиение данных***, которое я упомянул в конце предыдущей статьи.

```
from IPython.display import Image
Image(url= "imagefrom.png", width=500)
```

Изначально я сам думал, как машина вообще может обучаться на одних и тех же данных, которые мы ей даем? Ответ прост - мы пользуемся разбиением данных. Часть данных мы используем сначала для обучение, потом остальную часть для тестирования на величину ошибки, и тд. В идеале нужно проводить такой отбор несколько раз, находя лучшие коэффициенты и наименьшие ошибки, используя максимум данных. Так мы сможем наверняка сказать, есть ли у нас ошибка, или нет. Но как это делать быстро, а главное качественно? Давайте По порядку, для начала рассмотрим датасет, который я отковырял Здесь (на фотке выше как раз он и есть)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt # Визуализирует данные
import seaborn as sns # Визуализация данных в виде ступеньчатых данных и
тд
plt.rcParams['figure.figsize'] = [8,5]
plt.style.use('seaborn-v0_8')
df=pd.read_csv("climate_data.csv")
print(df)
```

```
             date    Tn    Tx  Tavg  RH_avg    RR   ss  ff_x  ddd_x  ff_avg
\
0      01-01-2010  21.4  30.2  27.1    82.0   9.0  0.5   7.0   90.0     5.0
1      02-01-2010  21.0  29.6  25.7    95.0  24.0  0.2   6.0   90.0     4.0
2      03-01-2010  20.2  26.8  24.5    98.0  63.0  0.0   5.0   90.0     4.0
3      04-01-2010  21.0  29.2  25.8    90.0   0.0  0.1   4.0  225.0     3.0
4      05-01-2010  21.2  30.0  26.7    90.0   2.0  0.4   NaN    NaN     NaN
...           ...   ...   ...   ...     ...   ...  ...   ...    ...     ...
```

```
589260  27-12-2020  25.2  31.2  29.2    74.0   0.0  1.4   4.0  280.0     2.0
589261  28-12-2020  25.3  31.6  28.1    78.0   NaN  3.0  12.0  260.0     2.0
589262  29-12-2020  24.6  32.3  28.4    81.0   NaN  6.5   5.0  260.0     2.0
589263  30-12-2020  25.2  32.6  28.4    80.0   0.0  2.4   7.0  260.0     2.0
589264  31-12-2020  24.3  32.0  26.7    86.0  26.6  5.8   7.0  350.0     2.0

       ddd_car  station_id
0            E       96001
1            E       96001
2            E       96001
3           SW       96001
4          NaN       96001
...        ...         ...
589260       C       97980
589261       C       97980
589262      SW       97980
589263       C       97980
589264       C       97980

[589265 rows x 12 columns]
```

Теперь давайте обсудим эти данные. Это климатические данные, взятые из Индонезии за 2010-2020 года. Всего список состоит из таких столбов, как:

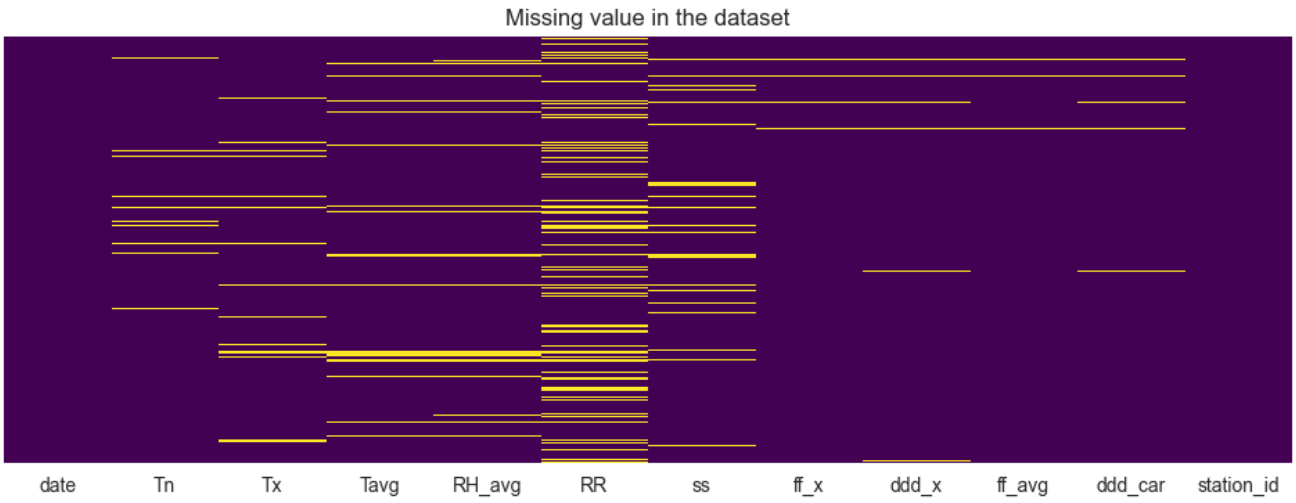
Сокращения

- date : дата взятие данных
- Tn : Минимальная температура (В Цельсиях)
- Tx : Максимальная температура (В Цельсиях)
- Tavg : Средняя температура
- RH_avg : Средняя влажность
- RR : Выпадение осадков (в мм)
- ss : Продолжительность ясной погоды (в часах)
- ff_x : Максимальная скорость воздушного потока (в м/с)
- ddd_x : Направление ветра при максимальной температура (в градусах)
- ff_avg : Средняя скорость воздушного потока (м/с)
- ddd_car : Среднее направление ветра (в градусах)
- station_id : id записывающей станции

Ради упрощения нашей задачи, опустим столбец с id станциями, скрестим пальцы, что id будет иметь низкую корреляцию с соответствующими данными

```
plt.figure(figsize=(12,4))
sns.heatmap(df.isnull(),cbar=False,cmap='viridis',yticklabels=False)
```

```
#разобраться
plt.title('Missing value in the dataset');
#sns.lmplot(x='RR',y='Tx',data=df,aspect=2,height=6)
```



Как видите, каждая желтая полоска обозначает пропущенное значение. Нам их для начала стоит убрать, потому что это не полные данные, у нас и так достаточно строк (500к+)

```
print("При первоначальном осмотре данных было выяснено, что некоторые данные о количество солнечных часов выходит за рамки 24-часового дня, пример ниже")
Image(url= "defect.png", width=800)
```

```
При первоначальном осмотре данных было выяснено, что некоторые данные о количество солнечных часов выходит за рамки 24-часового дня, пример ниже
```

```
print("Также пр осмотре было выявлено, что дни, когда солнечных часов за день накапливается меньше 1 часа, можно не рассматривать в работе из-за больших разбросов на всём промежутке")
Image(url= "defect_2.png", width=800)
```

```
Также пр осмотре было выявлено, что дни, когда солнечных часов за день накапливается меньше 1 часа, можно не рассматривать в работе из-за больших разбросов на всём промежутке
```

```
df=df.dropna(how='any') # Убираем все пустые строки
print("Поэтому было решено еще раз обработать модель, и все данные с солнечными часами, большими, чем 12 часов, было решено убрать, из-за наличия ночного времени")
print(f"Размер данных до убирания экстремальных значений {df.shape}")
df=df[df.ss<12]
print(f"Обработанный размер данных {df.shape}")
df=df[df.ss>1]
print(f"После исключения 1 солнечного часа в днях {df.shape}")
```

```
Поэтому было решено еще раз обработать модель, и все данные с солнечными часами, большими, чем 12 часов, было решено убрать, из-за наличия ночного времени
Размер данных до убирания экстремальных значений (372151, 12)
Обработанный размер данных (372027, 12)
После исключения 1 солнечного часа в днях (314264, 12)
```

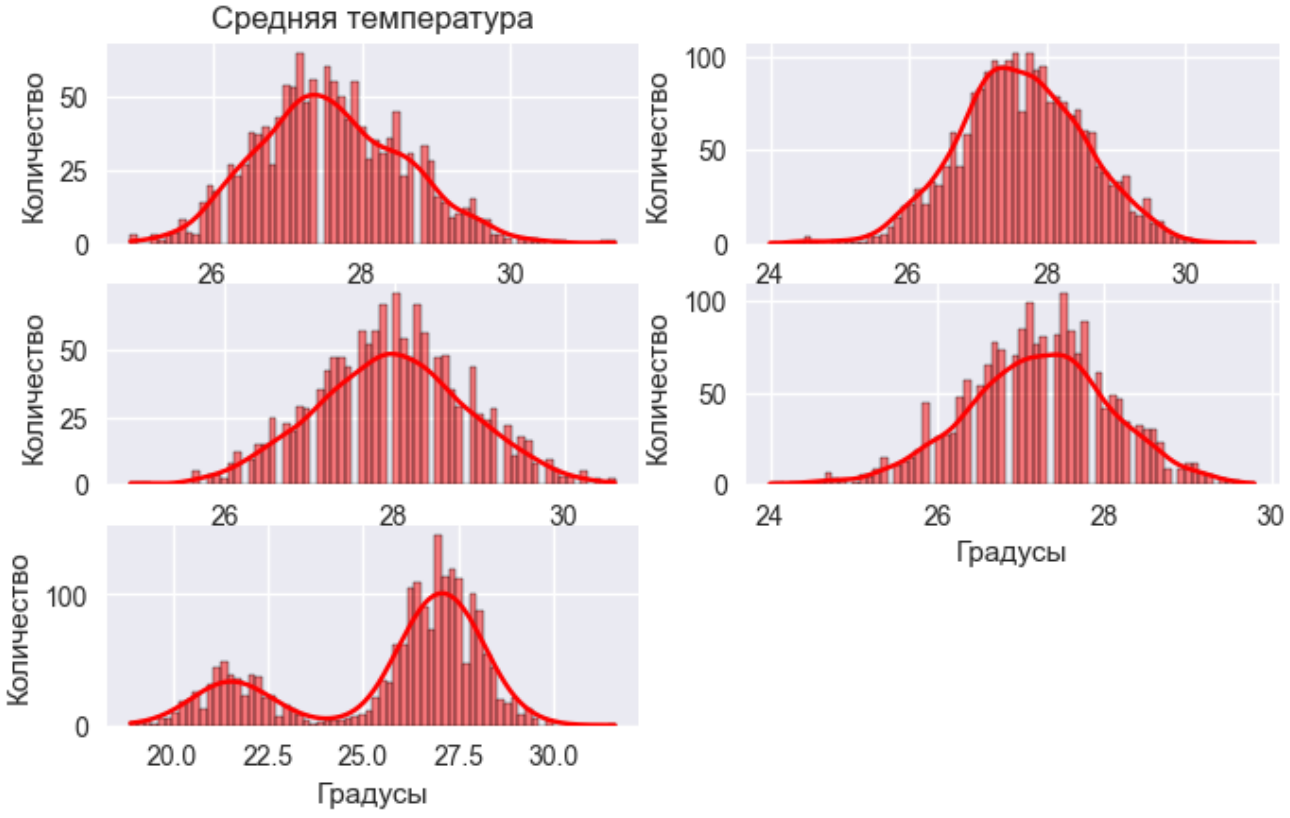
```
checker=0
r=-1
stations=list(df["station_id"])
ider=[]
counter=[]
for i in range(len(stations)):
    if stations[i]!=checker:
        checker=stations[i]
        ider.append(stations[i])
        r+=1
        counter.append(1)
    else:
        counter[r]+=1
#print(stations)
#print(ider)
#print(f"Количество встречаний id \n{counter}")
que = pd.DataFrame({
     'station_id': ider,
     'Count': counter,
})
print(que)
```

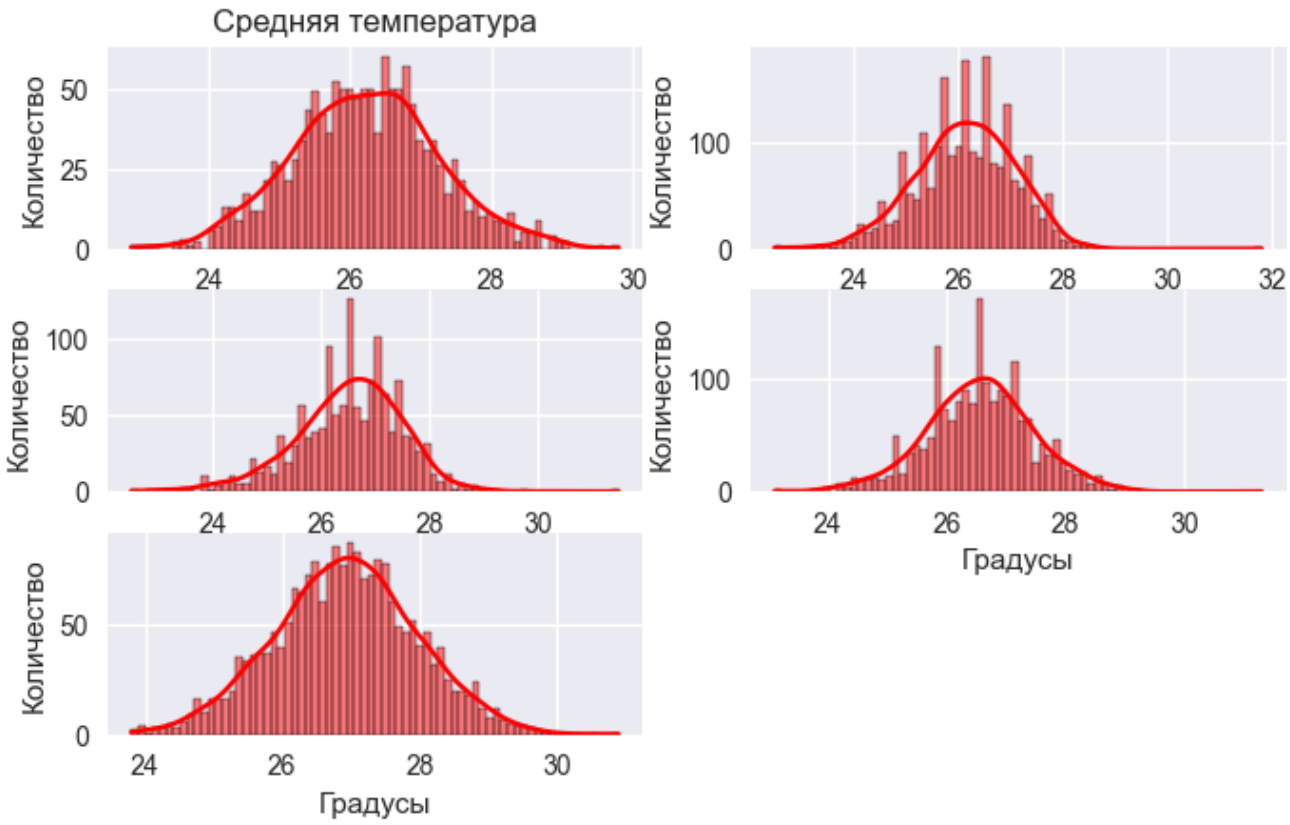
```
    station_id  Count
0         96001   1323
1         96009   2181
2         96011   1331
```

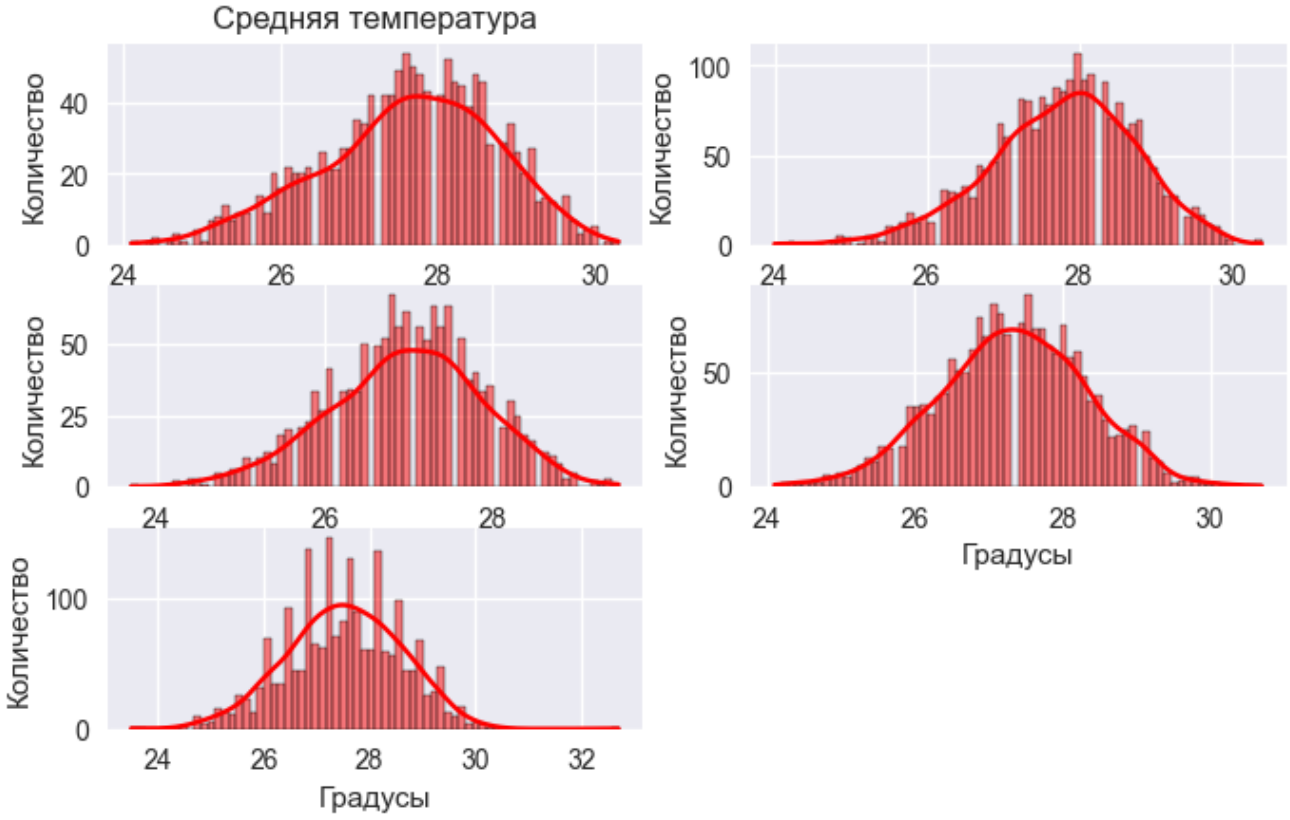
```
3         96015   1852
4         96017   2044
..          ...    ...
156       97796   2200
157       97810   1699
158       97876    961
159       97900   1913
160       97980   1145

[161 rows x 2 columns]
```

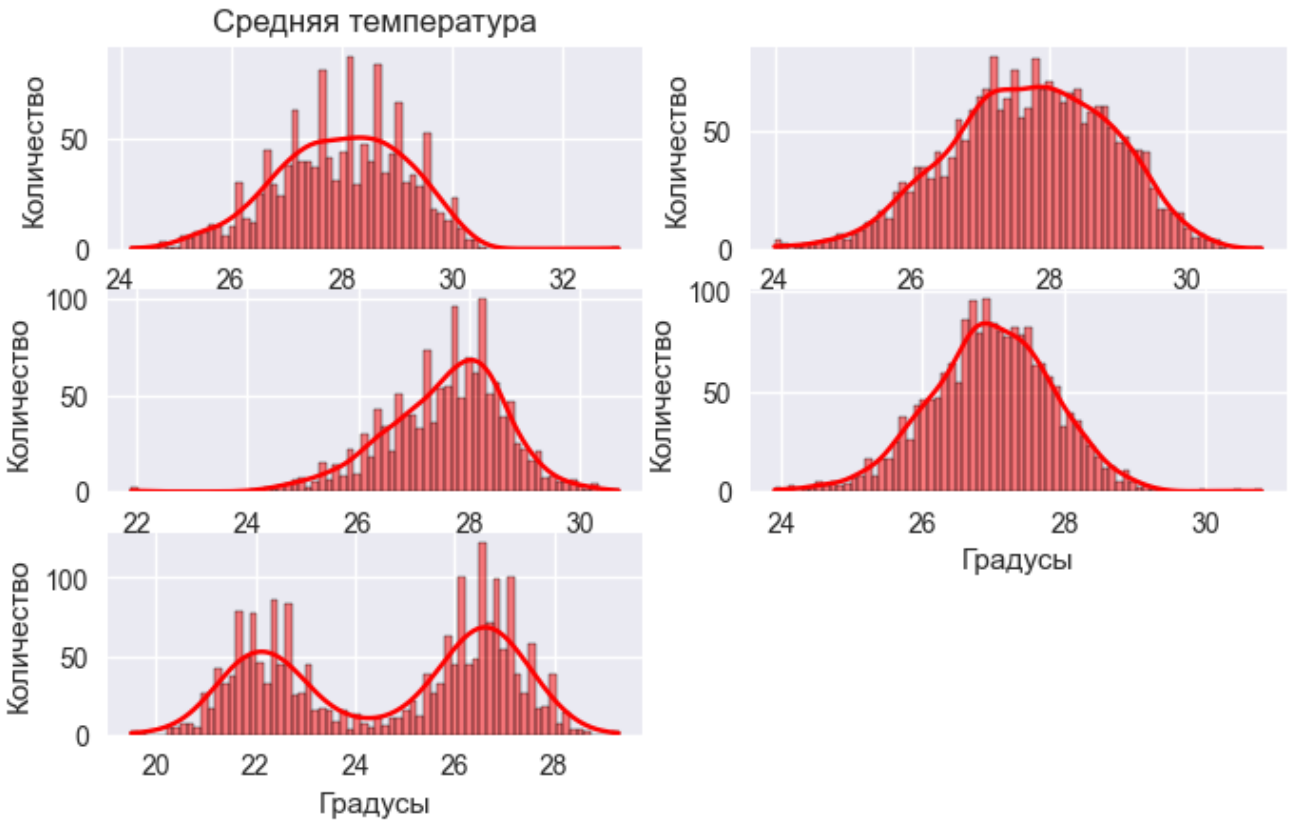
```
main=5
number_1=5
number_2=2
d=0
#d=que["Count"][50]
import warnings
warnings.filterwarnings('ignore')

for q in range(main):
    f= plt.figure(figsize=(8,8))
    for i in range(1,number_1+1):
        sub=100*number_1+10*number_2+i*1
        #print(d,"\n", df["station_id"][d:(d+que["Count"][i-1])],"\n" )
        ax=f.add_subplot(sub) # добавить
        sns.histplot(df['Tavg'][d:(d+que["Count"][i-
1])],bins=70,color='r',ax=ax,kde=True)
        if i==1:
            ax.set_title('Средняя температура')
        d+=que["Count"][i-1]
        ax.set_xlabel('Градусы')
        ax.set_ylabel('Количество')
```


Средняя температура
Количество
Градусы


Средняя температура
Количество
Градусы

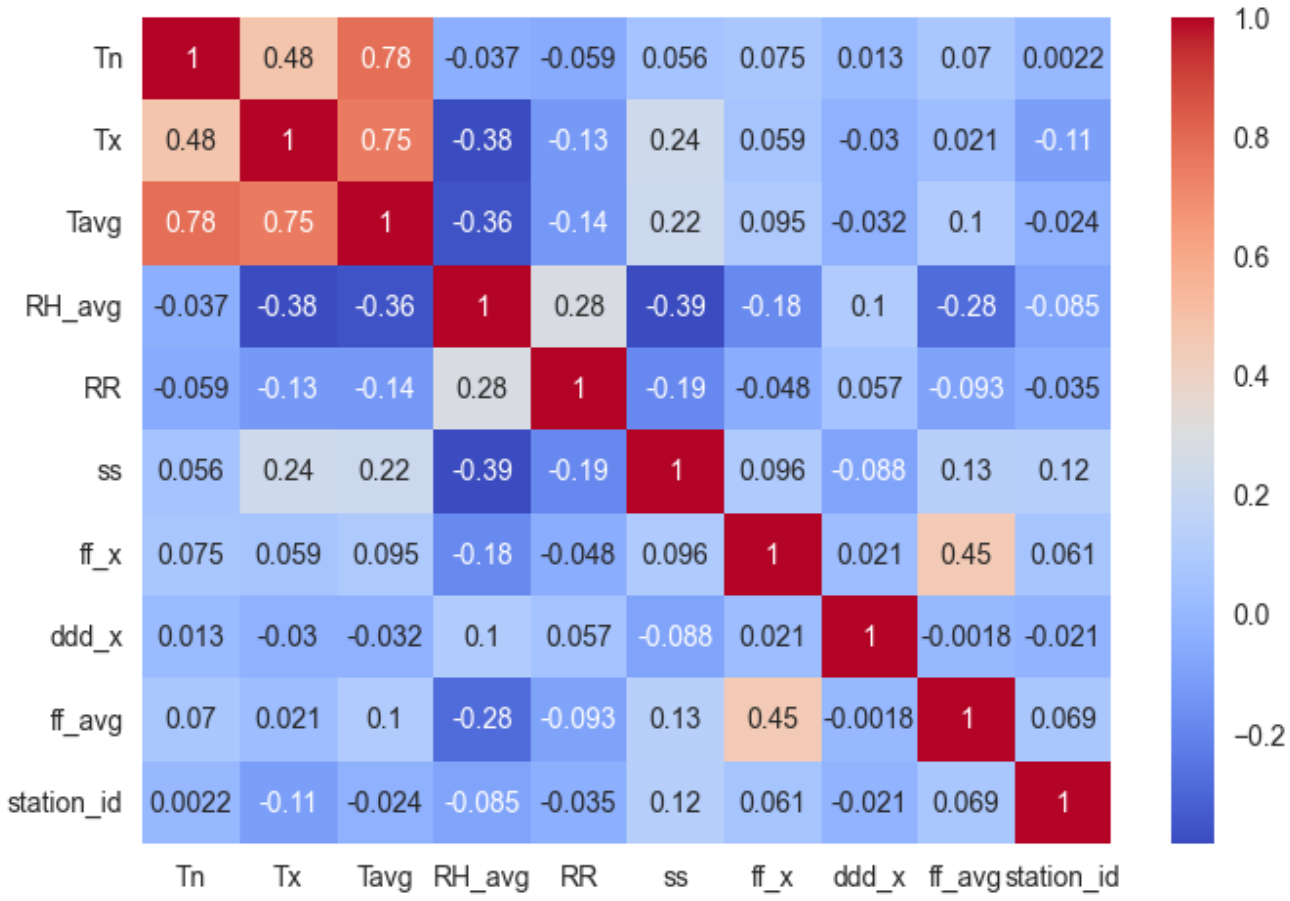

Средняя температура
Количество
Градусы


Средняя температура
Количество
Градусы



Как видите, мы отбросили все строки, где есть хотя бы одно нахождение NaN слова, и заодно уменьшили количество данных в полтора раза. По итогу у нас получилось так, что оказывается в датасете подряд идут данные со 162 различных датчиков, расположенных по всей Индонезии, поэтому сначала необходимо было убедиться, что у разных датчиков различия некоторых параметров не сильно отличается от региона нахождения, поэтому я создал небольшой список, в котором находится информация о количестве целых наблюдений, и, собственно, названия этих датчиков. В целом из-за того, что Индонезия не такая уж и большая страна, то датчики в целом показывают одинаковую среднюю температуру в разных регионах страны : примерно 27.5 Градусов. Давайте также посмотрим корреляционные тепловые карты, и также убедимся в этом:

```
corr = df.corr() # почитать что делает
sns.heatmap(corr, cmap = 'coolwarm', annot= True); #annot= дает числа в
квадратах, cmap = стиль карты
```



Конечно, пока что мы обработали наши данные, поэтому на этом этапе узнать о влиянии региона на температуру точно нельзя, но мне кажется, что эти зависимости хоть и будут присутствовать, но не так сильно, как если бы мы мерили датчики в той же России. Зато по карте отлично видно, что:

1. Средняя температура зависит от количество солнечных часов
2. Средняя скорость ветра сильно зависит от появлений максимальной скорости ветра.
3. Температурные данные сильно коррелируют, лучше использовать только одно значение Tavg
4. На удивление, средняя температура воздуха не слабо зависит от скорости ветра. Это может обозначать только то, что на их широте относительно стабильные ветра, да и цунами в таких областях, скорее всего, редко возникают.
5. Чем больше влажность воздуха, тем меньше вредняя скорость воздуха (что?). Могу только предположить, что это просто тропики, поэтому там поддерживается постоянная влажность из-за окружения морями со всех сторон (или возможно просто я глупый)

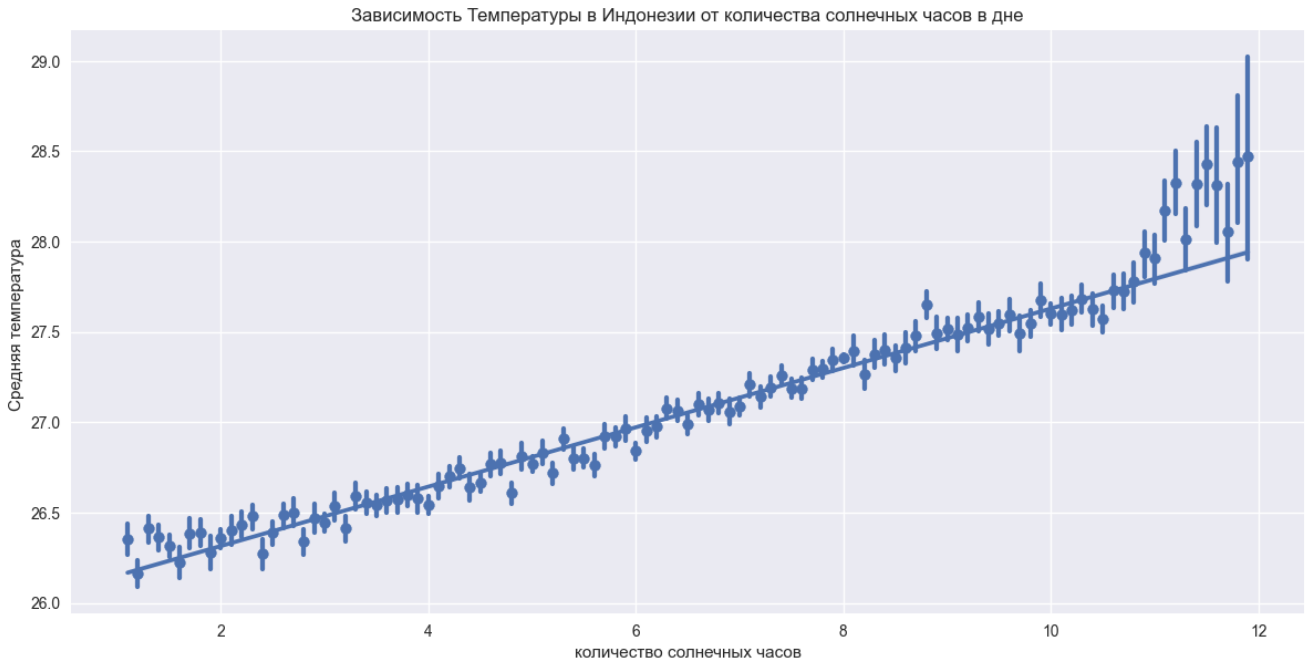
```
df_edited=df.drop(["station_id", "Tn", "Tx","ff_x","ddd_x","ddd_car",
"date"], axis=1)
print('Значения таблицы в изначальном датасете:\n',df.columns.values)
print('\nРазмер таблицы в изначальном датасете:',df.shape)
print('\nЗначения таблицы в датасете dummy:\n',df_edited.columns.values)
print('\nРазмер таблицы в измененном датасете:',df_edited.shape)
```

```
Значения таблицы в изначальном датасете:
 ['date' 'Tn' 'Tx' 'Tavg' 'RH_avg' 'RR' 'ss' 'ff_x' 'ddd_x' 'ff_avg'
 'ddd_car' 'station_id']

Размер таблицы в изначальном датасете: (314264, 12)

Значения таблицы в датасете dummy:
 ['Tavg' 'RH_avg' 'RR' 'ss' 'ff_avg']

Размер таблицы в измененном датасете: (314264, 5)
```

Я решил удалить ненужные для нас данные, потому что всё равно они нигде не будут сегодня использоваться (угол ветра), или же высокая корреляция делает эти данными бесполезными (максимальная температура). Также тк теперь мы записали в список que данные об id датчиков, они нам также не нужны. Теперь давайте посмотрим на оставшиеся данные, а именно посмотрим на зависимости всех оставшихся друг от друга

```
sns.lmplot(x='ss',y='Tavg',data=df_edited,aspect=2,height=6,
x_jitter=0.05, x_estimator=np.mean) # x_estimator отображает только
средние значения
plt.xlabel('количество солнечных часов')
plt.ylabel('Средняя температура ')
plt.title('Зависимость Температуры в Индонезии от количества солнечных
часов в дне')
```

```
Text(0.5, 1.0, 'Зависимость Температуры в Индонезии от количества солнечных
часов в дне')
```



Оппа!

Я не ожидал, но тут что-то в линейную зависимость выливается! А значит, давайте попробуем обучить модель на 80% данных датасета, и потом на оставашихся 20% попытаемся узнать, а точнее предугудать температуру в зависимости от количества солнечных часов! Было бы прекрасно, только представьте, создать машину, которая только по количеству яркости на улице определяла бы температуру в регионе. Конечно, будет некая погрешность, потому что датчики находили в разных частях страны, построящей из десятков островов, разброшанных по невероятно огромной терриории

Тренировка машинного обучения

Почитать, что такое тренировка у машины, можно, тыкнув левой кнопкой мыши вот по ЭТОМУ слову

```
from sklearn.model_selection import train_test_split
print(df_edited.columns.values)
# X = df_edited.drop(['Tavg'],axis=1) # Если не хотим убирать зависимость
всех переменных вместо одной
X = df_edited.drop(['Tavg','RH_avg','RR','ff_avg'],axis=1)# Убрали
зависимую переменную
y = df_edited['Tavg'] # отдельно запишем зависимую переменную
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.2,random_state=29)
print(X_train.shape,X_test.shape)
```

```
['Tavg' 'RH_avg' 'RR' 'ss' 'ff_avg']
(251411, 1) (62853, 1)
```

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_train,y_train)
y_predict=lin_reg.predict(X_test)
print(y_predict)
```

```
[26.93790568 27.13476033 27.03633301 ... 27.21678311 26.85588291
 26.38015083]
```

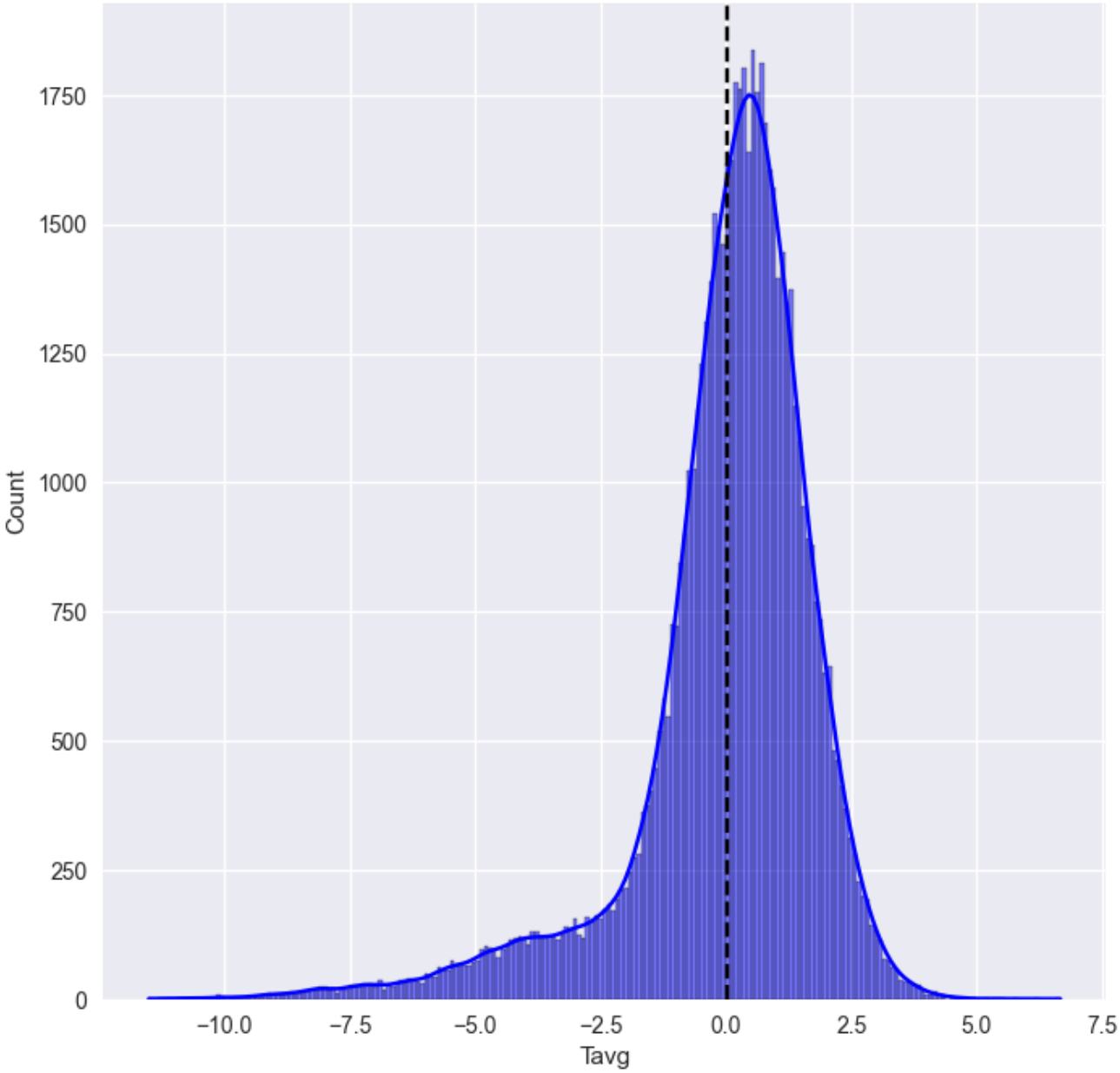
Оценим качество нашей модели. Для этого сначала оценить квадрат ошибки на тех данных, которые использовались для тренировки машины, и потом сверим их с ошибкой данных для теста (валидации):

```
y_predict_train=lin_reg.predict(X_train)
from sklearn.metrics import mean_squared_error
print("Качество на валидации: %.3f" %
mean_squared_error(y_test,y_predict))
print("Качество на тренировке: %.3f" %
mean_squared_error(y_train,y_predict_train))
print("Вау) Обычно качество на тестировочных данных должно быть хуже, чем
у тренировочных, потому что по последним обучалась машина. Однако у нас
это совсем не так))")
print("Но ошибка всё равно довольно большая, видимо некоторые из данных
слабо влияют на температуру.")
```

```
Качество на валидации: 3.274
Качество на тренировке: 3.346
Вау) Обычно качество на тестировочных данных должно быть хуже, чем у
тренировочных, потому что по последним обучалась машина. Однако у нас это
совсем не так))
Но ошибка всё равно довольно большая, видимо некоторые из данных слабо влияют
на температуру.
```

```
f= plt.figure(figsize=(8,8))
ax= f.add_subplot(111)
sns.histplot((y_test - y_predict),ax=ax,color='b', kde=True )
ax.axvline((y_test - y_predict).mean(),color='k',linestyle='--')
```

```
<matplotlib.lines.Line2D at 0x2c749af8a90>
```



Вывод

Хоть среднее значение отклонения и стоит в нуле, что обозначает правильную выборку тренировочных данных, отклонения от реальности в основном достигают значений в час, что уже хорошо, но что есть где-то 10% вероятность, что мы ошибемся на часов так 4-6, не дает мне покоя, но ладно

Проверка

Шел как-то Василий по Индонезии, и запомнил, что весь день светило яркое Солнце, он даже посчитал, что было 8.2 часа точно. Помогите Василию по этим данным определить, какая была температура воздуха!

Решение:

```
vas_ss=8.2
ans=lin_reg.coef_*vas_ss+lin_reg.intercept_
word="\\C^{\circ}"
print(f"Температура в Индонезии при Василии : %.1f C"  %ans)
```

```
Температура в Индонезии при Василии : 27.3 C
```

Конец!

Сегодня мы изучили метод разбиения

```

```

```

```

```

```