Trabalho Algebra Linear Computacional - Previsão Sono

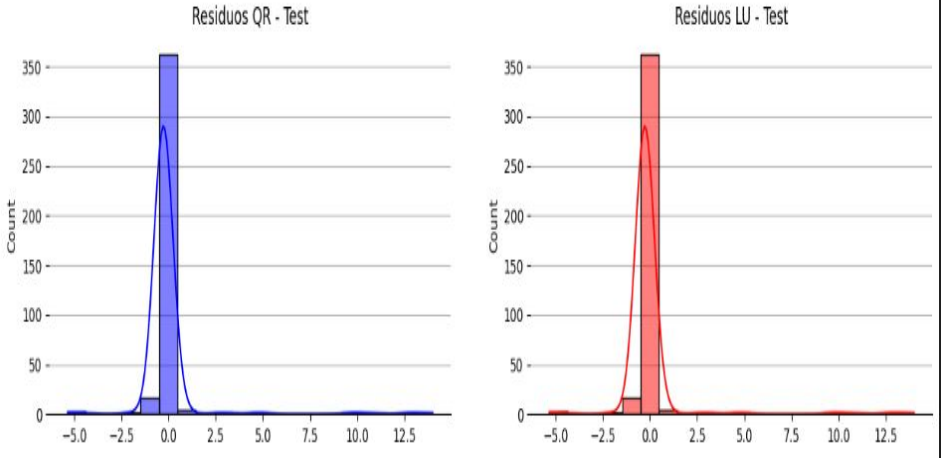
Mateus Sousa do Carmo - 495644

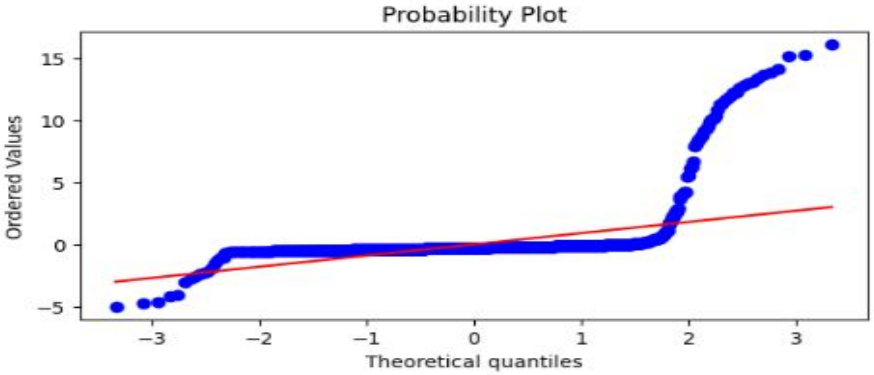
Modelos Utilizados

```python
def decomposicaoLU(M):
    A = np.dot(M.T, M)
    L = np.diag(np.ones(A.shape[1]))
    U = np.copy(L)

    for i in range(A.shape[0]):
        for j in range(A.shape[1]):
            if i<=j:
                U[i, j] = A[i, j] - np.sum(L[i, :j] * U[:j, j])
            if i >j:
                L[i, j] =  (A[i, j] - np.sum(L[i, :i] * U[:i, j])) / U[j, j]
    return L, U

def regressaoLU(M, b):
    L, U = decomposicaoLU(M)
    y = np.dot(M.T, b)
    x = np.linalg.solve(L, y)
    x = np.linalg.solve(U, x)
    return x
```

```python
def decomposicaoQR(A):
    row, col = A.shape
    if col > row: return 0

    # Inicializando as matrizes Q e R
    Q = np.zeros(shape=(row, col), dtype=np.float32)
    R = np.zeros(shape=(col, col), dtype=np.float32)
    u = 0
    for i in range(col):
        u = A[:, i]
        for j in range(i):
            R[j, i] = np.dot(Q[:, j], A[:, i])
            u -= np.multiply(R[j, i] , Q[:, j])

        R[i, i] = np.linalg.norm(u).astype(np.float32)
        if R[i, i] == 0: return 0
        Q[:, i] = np.divide(u , R[i, i]).astype(np.float32)
    return Q, R

def regressaoQR(M, b):
    M_ = M.copy()
    Q, R = decomposicaoQR(M_)

    alpha = np.dot(Q.T, b)

    norm_ = np.linalg.norm(np.dot(Q,Q.T) - np.eye(Q.shape[0]), "fro"
    print("Norm fro : ", norm_)
    return np.linalg.solve(R, alpha)
```

Resultados



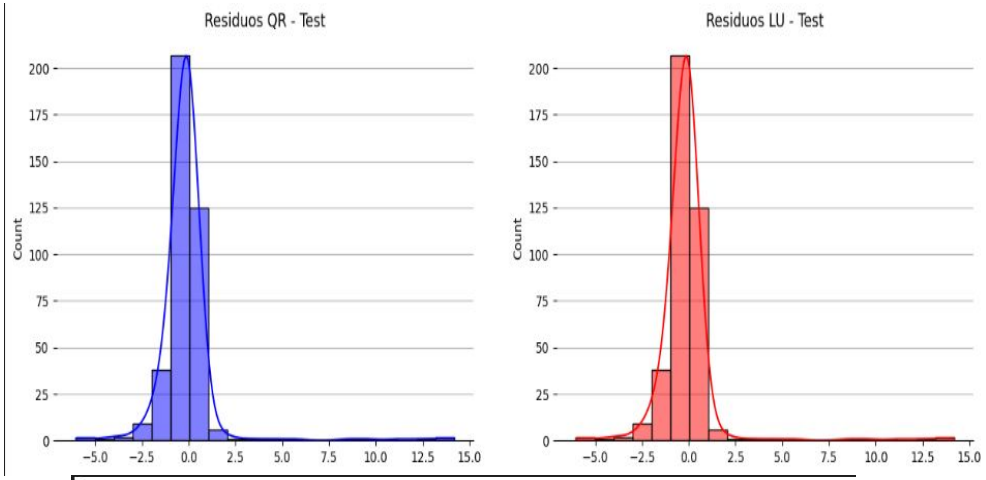


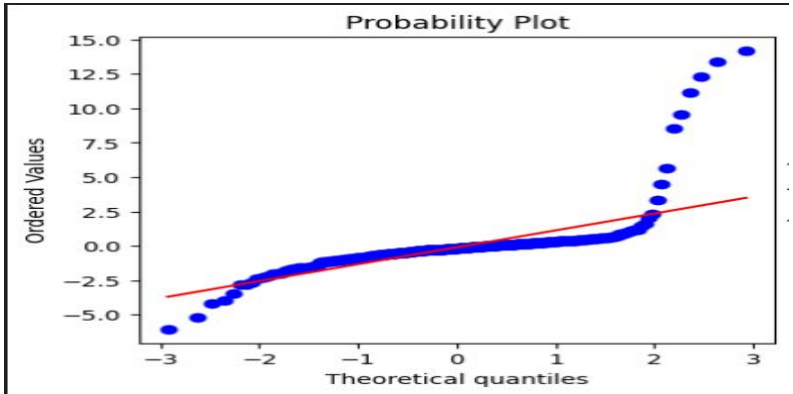
1. Os Resíduo não seguem uma distribuição normal, pode indicar:
   - Presença de outliers no conj. dados
   - Heterocedasticidade (variância não constante)
   - baixa qualidade nos dados.

2. Métricas tiveram o mesmo resultado.

```
Metricas para o método  LU
RMSE...:  3.160119925602487
MAE....:  0.5925617062897593
R2.....:  0.2465419888471131
```

Aplicando transformando modelo para tipo polinomial





- Métricas Piores RMSE: 2.97 e R2: 0.21
- Resíduos Não seguem uma distribuição normal

```
Metricas para o método  LU
RMSE...:  2.9797232526683386
MAE....:  0.7495953893043389
R2.....:  0.21231786338707148
```

Função abaixo transforma em polinomial do 2 grau

```
x = PolynomialFeatures(degree=4, include_bias=False).fit_transform(X)
x = np.concatenate((np.ones((x.shape[0], 1)), x), axis=1)
```

Comparando modelo

	QR	LU	QR_Poly	LU_Poly
Tempo de execução	0.06478	0.001807	0.708068	0.558634

- Método mais rápido foi o LU
- Método mais lendo QR

Conclusão

- Modelos Ruins
- Baixa qualidade dos dados
- “ O conjunto de dados foi gerado com base em uma equação matemática que simula como os fatores do estilo de vida influenciam a duração do sono. Inclui valores discrepantes para tornar os modelos robustos a dados ruidosos do mundo real”
- Melhor modelo LU no quesito tempo computacional