classification

December 11, 2023

Previsão de obesidade (Regressão Logística, SVM e MLP)

10/12/2023

Brenno de Oliveira da Rosa - 2021029935 Eduardo Alves Carvalho - 2021017550 Lucas Luan Belarmino Barbosa - 2021017872

1. Inicialmente, chamamos as importações que serão utilizadas no decorrer do código

[45]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.multiclass import OneVsRestClassifier
```

2. Após isso, adicionamos a tabela de dados relacionados com nossa pesquisa

[46]:
```
data = pd.read_csv('obesidade/Obesity Classification.csv')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 108 entries, 0 to 107
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      108 non-null    int64
 1   Age     108 non-null    int64
 2   Gender  108 non-null    object
 3   Height  108 non-null    int64
 4   Weight  108 non-null    int64
 5   BMI     108 non-null    float64
 6   Label   108 non-null    object
dtypes: float64(1), int64(4), object(2)
memory usage: 6.0+ KB
```

3. Aqui um exemplo dos primeiros dados da tabela:

[47]:
```
data.head()
```

[47]:

	ID	Age	Gender	Height	Weight	BMI	Label
0	1	25	Male	175	80	25.3	Normal Weight
1	2	30	Female	160	60	22.5	Normal Weight
2	3	35	Male	180	90	27.3	Overweight
3	4	40	Female	150	50	20.0	Underweight
4	5	45	Male	190	100	31.2	Obese

4. Após isso, definimos que analisaríamos a altura e o peso de cada um dos dados, juntamente com o nível de obesidade do indivíduo registrado. Dividimos os dados em abas de treino e teste

[75]:
```
from sklearn.preprocessing import StandardScaler
x = data[['Height','Weight']]  # Variáveis preditoras
y = data['Label']  # Variável alvo

scaler = StandardScaler()
x_normalized = scaler.fit_transform(x)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=42)
```

5. Criamos os modelos de classificação e treinamos com os dados de treino.

[73]:
```
# Regressão Logística
logreg_model = LogisticRegression()
logreg_model.fit(x_train, y_train)
logreg_predictions = logreg_model.predict(x_test)

# SVM
svm_model = SVC()
svm_model.fit(x_train, y_train)
svm_predictions = svm_model.predict(x_test)

# MLP
mlp_model = MLPClassifier()
mlp_model.fit(x_train, y_train)
mlp_predictions = mlp_model.predict(x_test)
```

```
C:\Users\Usuario\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz
5n2kfra8p0\LocalCache\local-packages\Python311\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\Users\Usuario\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz
5n2kfra8p0\LocalCache\local-packages\Python311\site-
packages\sklearn\neural_network\_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

6. Verificamos a acurácia de cada um dos métodos que utilizariamos para ver se o modelo escolhido é valido.

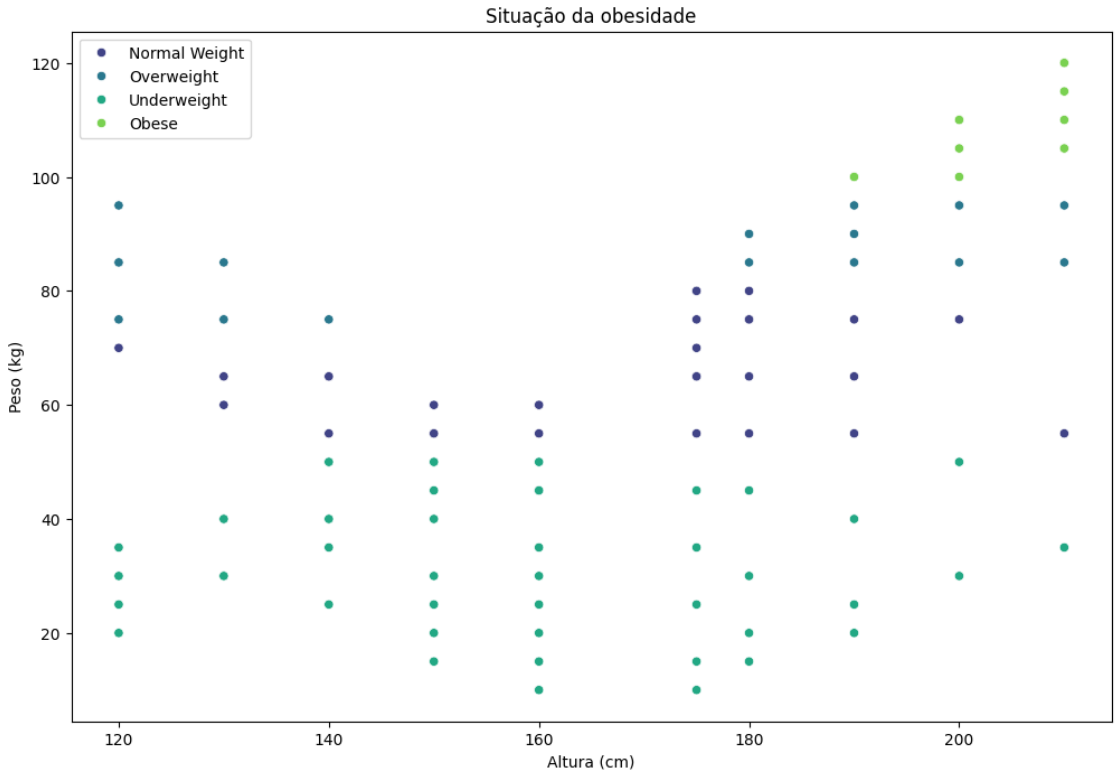
[63]:
```
from sklearn.svm import LinearSVC



accuracy_logreg = accuracy_score(y_test, logreg_predictions)
accuracy_svm = accuracy_score(y_test, svm_predictions)
accuracy_mlp = accuracy_score(y_test, mlp_predictions)

print(f"Acurácia Regressão Logística: {accuracy_logreg}")
print(f"Acurácia SVM: {accuracy_svm}")
print(f"Acurácia MLP: {accuracy_mlp}")

# SVC
svc = SVC()

# MPL
mpl = OneVsRestClassifier(LinearSVC())

# Regressão Logística
logreg = LogisticRegression()
```

```
Acurácia Regressão Logística: 0.8181818181818182
Acurácia SVM: 0.8181818181818182
Acurácia MLP: 0.6818181818181818
```

Como a acurácia foi de 0.818181..., então continuamos neste modelo. 7. Imprimimos um gráfico para analisarmos as informações:

[64]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 8))
sns.scatterplot(x='Height', y='Weight', hue='Label', data=data,␣
 ↪palette='viridis')

plt.xlabel('Altura (cm)')
```

```
plt.ylabel('Peso (kg)')
plt.title('Situação da obesidade')

plt.legend(loc='upper left')

plt.show()
```

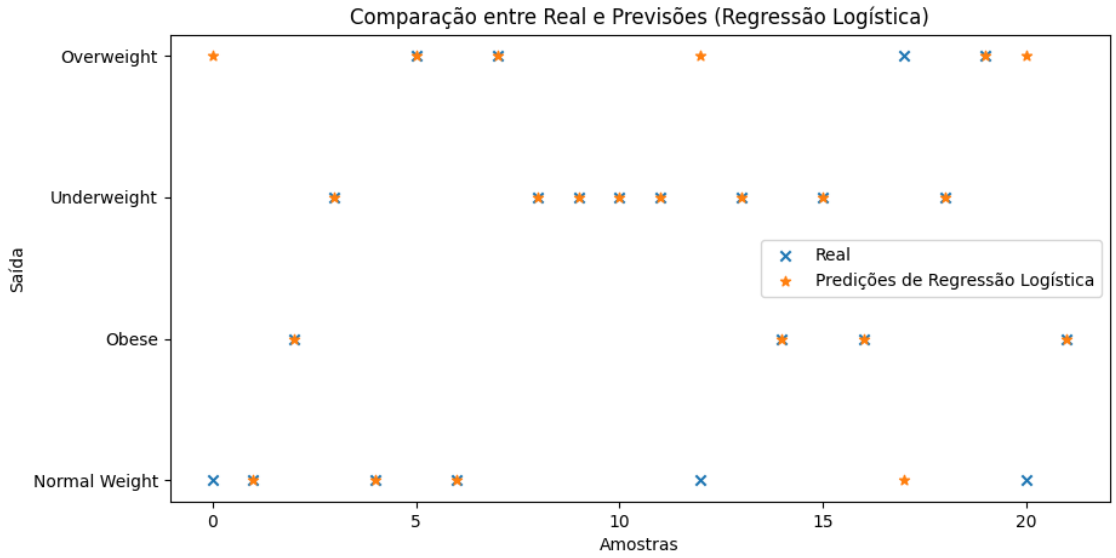


Pelo gráfico, é possível confirmar que o nível de obesidade dos indivíduos está diretamente relacionada com sua altura e peso.

8. Realizando as predições através da regressão logística:

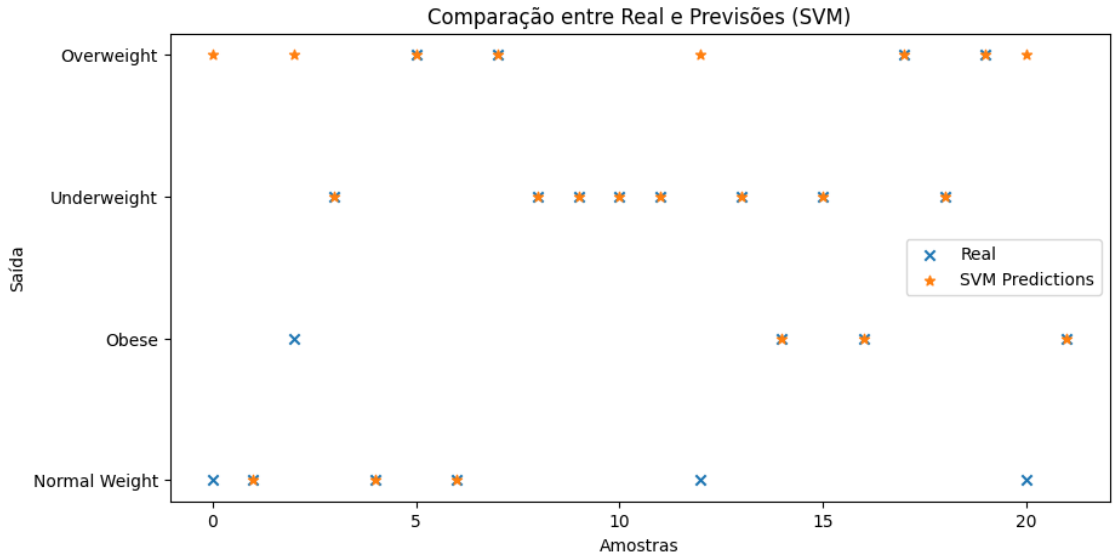
[68]:
```
plt.figure(figsize=(10, 5))

plt.scatter(range(len(y_test)), y_test, marker='x', label='Real')

# Plotar previsões do modelo de Regressão Logística
plt.scatter(range(len(logreg_predictions)), logreg_predictions, marker='*',␣
 ↪label='Predições de Regressão Logística')

plt.legend()
plt.xlabel('Amostras')
plt.ylabel('Saída')
```

```
plt.title('Comparação entre Real e Previsões (Regressão Logística)')
plt.show()
```



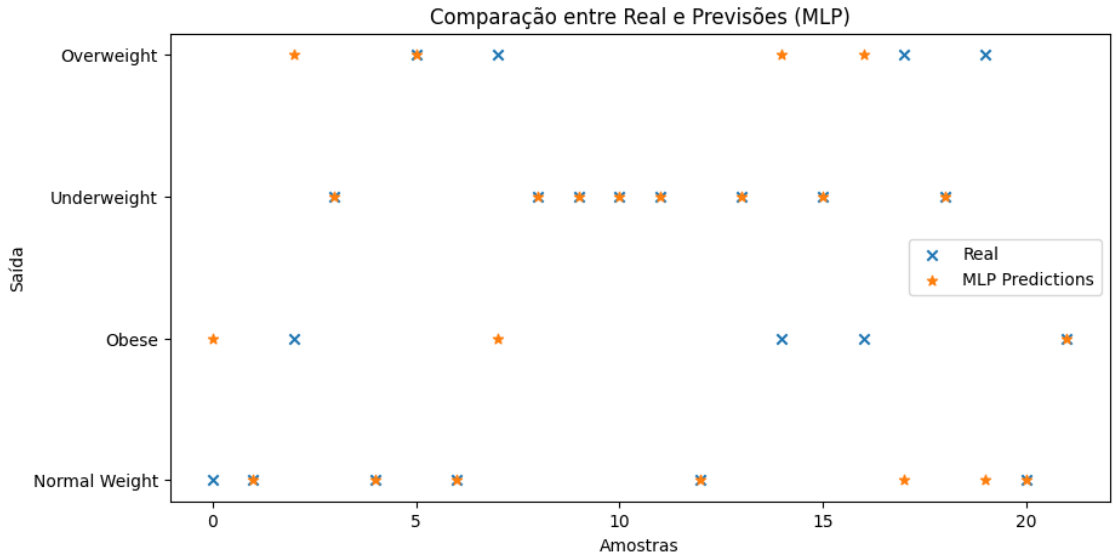
9. Realizando a predição através do modelo SVM:

[69]:
```
plt.figure(figsize=(10, 5))

plt.scatter(range(len(y_test)), y_test, marker='x', label='Real')

# Plotar previsões do modelo SVM
plt.scatter(range(len(svm_predictions)), svm_predictions, marker='*',␣
 ↪label='SVM Predictions')

plt.legend()
plt.xlabel('Amostras')
plt.ylabel('Saída')
plt.title('Comparação entre Real e Previsões (SVM)')
plt.show()
```



10. Realizando a predição através do modelo MLP:

[70]:
```
#plotar gráfico
plt.figure(figsize=(10, 5))

# Plotar saídas reais
plt.scatter(range(len(y_test)), y_test, marker='x', label='Real')

# Plotar previsões do modelo MLP
plt.scatter(range(len(mlp_predictions)), mlp_predictions, marker='*',␣
 ↪label='MLP Predictions')

plt.legend()
plt.xlabel('Amostras')
plt.ylabel('Saída')
plt.title('Comparação entre Real e Previsões (MLP)')
plt.show()
```



Podemos analisar pelos gráficos que o MLP possui a menor acurácia em comparação com o SVM e o de regressão logística.

Todos demonstram ser bem eficazes, considerando que a maioria das predições estavam de acordo com os valores reais. Existiram sim, alguns casos onde a predição não batia com o valor, mas em grande parte das amostrar a predição foi correta