



Predição de indivíduos com Depressão com Machine Learning

Apresentação do projeto final de Machine Learning

Lucas de Oliveira Telles
Erick Mathias Valu
Gabriel Vitor Cedran

Objetivo

Construir um modelo que faça a predição de indivíduos que possuam ou não depressão baseados em dados retirados de entrevistas de saúde mental.

O modelo deve analisar os dados disponibilizados da entrevista e definir se o individuo possui ou não depressão. A eficacia do modelo deve ser medida com o Accuracy Score.

Por fim, deve ser gerado um arquivo “submission.csv” com as definições do modelo para dados separados para prova.

Dados



Train.csv

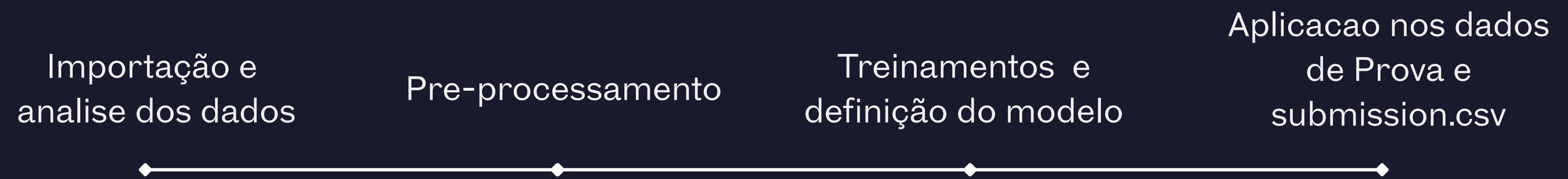
Dados para treinamento e teste de modelo



Test.csv

Dados para prova, resultados no submission.csv

Pipeline



Analise dos dados

Neste primeiro momento foi realizada a avaliação dos dados que vão ser utilizados para treinamento do modelo.

Para tal, foram realizados os seguintes passos:

- Importação dos datasets em dataframes do pandas
- Verificação das dimensões dos datasets
- Visualização dos datasets
- Analise das colunas e informações
- Verificação da quantidade de valores nulos por coluna

Analise dos dados

Realizando a importação dos datasets para dataframes do pandas

```
import numpy as np
import pandas as pd

train_data = pd.read_csv("data/train.csv")
test_data = pd.read_csv("data/test.csv")
```

✓ 0.7s

Analise dos dados

Verificação das dimensões dos datasets

```
train_data.shape
✓ 0.0s
(93800, 19)

test_data.shape
✓ 0.0s
(93800, 19)
```

Analise dos dados

Visualizando os datasets

```
# visualizacao da tabela de treino
train_data.head()
```

0.0s

```
# visualizacao da tabela de prova
test_data.head()
```

0.0s

					Working Professional or Student	Profession	Academic Pressure	Work Pressure	CGPA	Study Satisfaction	Job Satisfaction	Sleep Duration	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work
0	0	Aaradhyा	Female	49.0	Ludhiana	Working Professional	Chef	NaN	5.0	NaN	NaN	2.0	More than 8 hours	Healthy	BHM	No
1	1	Vivan	Male	26.0	Varanasi	Working Professional	Teacher	NaN	4.0	NaN	NaN	3.0	Less than 5 hours	Unhealthy	LLB	Yes
2	2	Yuvraj	Male	33.0	Visakhapatnam	Student	NaN	5.0	NaN	8.97	2.0	NaN	5-6 hours	Healthy	B.Pharm	Yes
3	3	Yuvraj	Male	22.0	Mumbai	Working Professional	Teacher	NaN	5.0	NaN	NaN	1.0	Less than 5 hours	Moderate	BBA	Yes
4	4	Rhea	Female	30.0	Kanpur	Working Professional	Business Analyst	NaN	1.0	NaN	NaN	1.0	5-6 hours	Unhealthy	BBA	Yes

					Working Professional or Student	Profession	Academic Pressure	Work Pressure	CGPA	Study Satisfaction	Job Satisfaction	Sleep Duration	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work
0	140700	Shivam	Male	53.0	Visakhapatnam	Working Professional	Judge	NaN	2.0	NaN	NaN	5.0	Less than 5 hours	Moderate	LLB	No
1	140701	Sanya	Female	58.0	Kolkata	Working Professional	Educational Consultant	NaN	2.0	NaN	NaN	4.0	Less than 5 hours	Moderate	B.Ed	No
2	140702	Yash	Male	53.0	Jaipur	Working Professional	Teacher	NaN	4.0	NaN	NaN	1.0	7-8 hours	Moderate	B.Arch	Yes
3	140703	Nalini	Female	23.0	Rajkot	Student	NaN	5.0	NaN	6.84	1.0	NaN	More than 8 hours	Moderate	BSc	Yes
4	140704	Shaurya	Male	47.0	Kalyan	Working Professional	Teacher	NaN	5.0	NaN	NaN	5.0	7-8 hours	Moderate	BCA	Yes

Analise dos dados

Visualizando as colunas e informações dos datasets

```
# mostrando as colunas presentes na tabela de treinamento
train_data.info()

✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 140700 entries, 0 to 140699
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               140700 non-null   int64  
 1   Name              140700 non-null   object  
 2   Gender             140700 non-null   object  
 3   Age                140700 non-null   float64 
 4   City               140700 non-null   object  
 5   Working Professional or Student 140700 non-null   object  
 6   Profession         104070 non-null   object  
 7   Academic Pressure  27897 non-null    float64 
 8   Work Pressure      112782 non-null   float64 
 9   CGPA               27898 non-null    float64 
 10  Study Satisfaction 27897 non-null    float64 
 11  Job Satisfaction   112790 non-null   float64 
 12  Sleep Duration     140700 non-null   object  
 13  Dietary Habits    140696 non-null   object  
 14  Degree              140698 non-null   object  
 15  Have you ever had suicidal thoughts ? 140700 non-null   object  
 16  Work/Study Hours   140700 non-null   float64 
 17  Financial Stress   140696 non-null   float64 
 18  Family History of Mental Illness 140700 non-null   object  
 19  Depression          140700 non-null   int64  
dtypes: float64(8), int64(2), object(10)
memory usage: 21.5+ MB
```

```
# mostrando as colunas presentes na tabela de prova
test_data.info()

✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 93800 entries, 0 to 93799
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               93800 non-null   int64  
 1   Name              93800 non-null   object  
 2   Gender             93800 non-null   object  
 3   Age                93800 non-null   float64 
 4   City               93800 non-null   object  
 5   Working Professional or Student 93800 non-null   object  
 6   Profession         69168 non-null   object  
 7   Academic Pressure  18767 non-null    float64 
 8   Work Pressure      75022 non-null   float64 
 9   CGPA               18766 non-null    float64 
 10  Study Satisfaction 18767 non-null    float64 
 11  Job Satisfaction   75026 non-null   float64 
 12  Sleep Duration     93800 non-null   object  
 13  Dietary Habits    93795 non-null   object  
 14  Degree              93798 non-null   object  
 15  Have you ever had suicidal thoughts ? 93800 non-null   object  
 16  Work/Study Hours   93800 non-null   float64 
 17  Financial Stress   93800 non-null   float64 
 18  Family History of Mental Illness 93800 non-null   object  
dtypes: float64(8), int64(1), object(10)
memory usage: 13.6+ MB
```

Analise dos dados

Visualizando a quantidade de valores nulos por coluna

```
# verificando a quantidade de valores nulos das features da tabela de treino  
train_data.isnull().sum()  
✓ 0.0s  


|                                       |              |
|---------------------------------------|--------------|
| id                                    | 0            |
| Name                                  | 0            |
| Gender                                | 0            |
| Age                                   | 0            |
| City                                  | 0            |
| Working Professional or Student       | 0            |
| Profession                            | 36630        |
| Academic Pressure                     | 112803       |
| Work Pressure                         | 27918        |
| CGPA                                  | 112802       |
| Study Satisfaction                    | 112803       |
| Job Satisfaction                      | 27910        |
| Sleep Duration                        | 0            |
| Dietary Habits                        | 4            |
| Degree                                | 2            |
| Have you ever had suicidal thoughts ? | 0            |
| Work/Study Hours                      | 0            |
| Financial Stress                      | 4            |
| Family History of Mental Illness      | 0            |
| Depression                            | 0            |
|                                       | dtype: int64 |


```

```
# verificando a quantidade de valores nulos das features da tabela de prova  
test_data.isnull().sum()  
✓ 0.0s  


|                                       |              |
|---------------------------------------|--------------|
| id                                    | 0            |
| Name                                  | 0            |
| Gender                                | 0            |
| Age                                   | 0            |
| City                                  | 0            |
| Working Professional or Student       | 0            |
| Profession                            | 24632        |
| Academic Pressure                     | 75033        |
| Work Pressure                         | 18778        |
| CGPA                                  | 75034        |
| Study Satisfaction                    | 75033        |
| Job Satisfaction                      | 18774        |
| Sleep Duration                        | 0            |
| Dietary Habits                        | 5            |
| Degree                                | 2            |
| Have you ever had suicidal thoughts ? | 0            |
| Work/Study Hours                      | 0            |
| Financial Stress                      | 0            |
| Family History of Mental Illness      | 0            |
|                                       | dtype: int64 |


```

Pre-processamento

Apos a visualização e entendimento dos dados, foi realizada a etapa de preparação desses mesmos dados para serem utilizados para treinamento dos modelos

Para essa finalidade, foram realizados os seguintes passos:

- Unificação de features
- Exclusão de features desnecessárias
- Preenchimento de valores nulos das features
- Transformação de features categóricas para numéricas
- Aplicação das tratativas nos datasets e visualização
- Escalonamento das features para treinamento dos modelo

Pre-processamento

Unificando Work/Academic Pressure e Study/Job Satisfaction em uma feature cada e exclusão de features desnecessárias para treinamento

```
def trata_features(df):
    """
    Tem a função de criar features e excluir as desnecessárias
    """

    # junta as colunas de Work Pressure e Academic Pressure em uma unica coluna Work/Academic Pressure
    df['Work/Academic Pressure'] = df.apply(
        lambda row:
            row['Academic Pressure'] if row['Working Professional or Student'] == 'Student'
            else row['Work Pressure'],
        axis=1
    )

    # junta as colunas de Study Satistaction e Job Satistaction em uma unica coluna Study/Job Satistaction
    df['Study/Job Satistaction'] = df.apply(
        lambda row:
            row['Study Satisfaction'] if row['Working Professional or Student'] == 'Student'
            else row['Job Satisfaction'],
        axis=1
    )

    # remove as colunas desnecessárias para o processo
    df = df.drop(['Academic Pressure', 'Work Pressure', 'Study Satisfaction', 'Job Satisfaction', 'id', 'City', 'Name'], axis=1)

    return df
```

Pre-processamento

Preenchimento de valores nulos das features

```
def preenche_dados(df):
    """
    Tem a função de preencher os dados vazios/faltantes da tabela
    """

    # cria uma cópia do dataframe
    df = df.copy()

    # preenche as colunas Financial Stress, Work/Academic Pressure, Study/Job Satisfaction com suas respectivas modas
    df.loc[:, 'Financial Stress'] = df['Financial Stress'].fillna(df['Financial Stress'].mode()[0])
    df.loc[:, 'Work/Academic Pressure'] = df['Work/Academic Pressure'].fillna(df['Work/Academic Pressure'].mode()[0])
    df.loc[:, 'Study/Job Satisfaction'] = df['Study/Job Satisfaction'].fillna(df['Study/Job Satisfaction'].mode()[0])

    # recuperando a média de CGPA
    cgpa_mean = df['CGPA'].mean()

    # aplica 0 em CGPA para indivíduos que trabalham já que não se aplica para eles
    df['CGPA'] = df.apply(
        lambda row:
            0 if row['Working Professional or Student'] == 'Working Professional'
            else row['CGPA'],
        axis=1
    )

    # preenche os espaços vazios com a média do CGPA
    df.loc[:, 'CGPA'] = df['CGPA'].fillna(cgpa_mean)

    # recupera os valores que mais aparecem em Degree
    degree = df['Degree'].value_counts()[0:27].reset_index()['Degree'].tolist()
    # percorre os registros, caso o valor do registro esteja em degree o valor é mantido
    # se não ele é substituído por Other
    df['Degree'] = np.where(df['Degree'].isin(degree), df['Degree'], "Other")

    # recupera os valores que mais aparecem em Profession
    profession = df['Profession'].value_counts()[0:35].reset_index()['Profession'].tolist()
    # percorre os registros, caso o valor do registro esteja em profession o valor é mantido
    # se não ele é substituído por Other
    df['Profession'] = np.where(df['Profession'].isin(profession), df['Profession'], "Other")

return df
```

Pre-processamento

Conversao de features categoricas para numericas e preenchimento de dados faltantes

```
def encode(df):
    """
    Tem a finalidade de transformar as features categoricas em numericas
    """

    # trata o campo Dietary Habits
    diet = {
        'More Healthy':0,
        'Healthy':1,
        '5 Healthy':1,
        'Less than Healthy':2,
        'Less Healthy':2,
        'Moderate':3,
        'Unhealthy':4,
        '5 Unhealthy':4,
        'No Healthy':4,
    }
    df['Dietary Habits'] = df['Dietary Habits'].apply(
        lambda valor: diet.get(valor, None)
    )

    # preenche os campos vazios com a moda
    df.loc[:, 'Dietary Habits'] = df['Dietary Habits'].fillna(df['Dietary Habits'].mode()[0])
```

Pre-processamento

Conversão de features categóricas para numéricas e preenchimento de dados faltantes

```
# trata o campo Sleep Duration
sleep = {
    "More than 8 hours":9,
    'Less than 5 hours':4,
    '5-6 hours':5.5,
    '7-8 hours':7.5,
    '1-2 hours':1.5,
    '6-8 hours':7,
    '4-6 hours':5,
    '6-7 hours':6.5,
    '10-11 hours':10.5,
    '8-9 hours':8.5,
    '9-11 hours':10,
    '2-3 hours':2.5,
    '3-4 hours':3.5,
    'Moderate':6,
    '4-5 hours':4.5,
    '9-6 hours':7.5,
    '1-3 hours':2,
    '1-6 hours':4,
    '8 hours':8,
    '10-6 hours':8,
    'Unhealthy':3,
    'Work_Study_Hours':6,
    '3-6 hours':3.5,
    '9-5':7,
    '9-5 hours':7,
}
df['Sleep Duration'] = df['Sleep Duration'].map(sleep)

# preenche os registros vazios com a moda
df.loc[:, 'Sleep Duration'] = df['Sleep Duration'].fillna(df['Sleep Duration'].mode()[0])
```

Pre-processamento

Conversão de features categóricas para numéricas e preenchimento de dados faltantes

```
# trata o campo Gender
df['Gender'] = df['Gender'].map({'Male':0, 'Female':1})

# trata o campo Have you ever had suicidal thoughts ?
df['Have you ever had suicidal thoughts ?'] = df['Have you ever had suicidal thoughts ?'].map({'No':0, 'Yes':1})

# trata o campo Working Professional or Student
df['Working Professional or Student'] = df['Working Professional or Student'].map({'Student':0, 'Working Professional':1})

# trata o campo Family History of Mental Illness
df['Family History of Mental Illness'] = df['Family History of Mental Illness'].map({'No':0, 'Yes':1})

# transforma as variaveis categoricas em numericas
label_encoder = LabelEncoder()
df['Profession'] = label_encoder.fit_transform(df['Profession'])
df['Degree'] = label_encoder.fit_transform(df['Degree'])

return df
```

Pre-processamento

Aplicando as funções tratativas nos datasets

```
# realiza o tratamento da tabela de treino
train_data = trata_features(train_data)
train_data = preenche_dados(train_data)
train_data = encode(train_data)

# realiza o tratamento da tabela de prova
test_data = trata_features(test_data)
test_data = preenche_dados(test_data)
test_data = encode(test_data)
```

Pre-processamento

Visualização dos dados apos as tratativas

```
# verifica se possui valores nulls  
train_data.isnull().sum()  
  
✓ 0.0s  
  
Gender 0  
Age 0  
Working Professional or Student 0  
Profession 0  
CGPA 0  
Sleep Duration 0  
Dietary Habits 0  
Degree 0  
Have you ever had suicidal thoughts ? 0  
Work/Study Hours 0  
Financial Stress 0  
Family History of Mental Illness 0  
Depression 0  
Work/Academic Pressure 0  
Study/Job Satistaction 0  
dtype: int64
```

```
# verifica se possui valores nulls  
test_data.isnull().sum()  
  
✓ 0.0s  
  
Gender 0  
Age 0  
Working Professional or Student 0  
Profession 0  
CGPA 0  
Sleep Duration 0  
Dietary Habits 0  
Degree 0  
Have you ever had suicidal thoughts ? 0  
Work/Study Hours 0  
Financial Stress 0  
Family History of Mental Illness 0  
Work/Academic Pressure 0  
Study/Job Satistaction 0  
dtype: int64
```

Pre-processamento

Visualização dos dados apos as tratativas

```
# visualizacao da tabela apos a tratativa
train_data.head()
```

Python

	Gender	Age	Working Professional or Student	Professional	CGPA	Sleep Duration	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Depression	Work/Academic Pressure	Study/Job Satisfaction	
0	1	49.0		1	3	0.00	9.0	1.0	9	0	1.0	2.0	0	0	5.0	2.0
1	0	26.0		1	33	0.00	4.0	4.0	12	1	7.0	3.0	0	1	4.0	3.0
2	0	33.0		0	25	8.97	5.5	1.0	3	1	3.0	1.0	0	1	5.0	2.0
3	0	22.0		1	33	0.00	4.0	3.0	6	1	10.0	1.0	1	1	5.0	1.0
4	1	30.0		1	2	0.00	5.5	4.0	6	1	9.0	4.0	1	0	1.0	1.0

```
# visualizacao da tabela apos a tratativa
test_data.head()
```

✓ 0.0s Python

	Gender	Age	Working Professional or Student	Professional	CGPA	Sleep Duration	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Work/Academic Pressure	Study/Job Satisfaction	
0	0	53.0		1	20	0.00	4.0	3.0	12	0	9.0	3.0	1	2.0	5.0
1	1	58.0		1	12	0.00	4.0	3.0	2	0	6.0	4.0	0	2.0	4.0
2	0	53.0		1	33	0.00	7.5	3.0	0	1	12.0	4.0	0	4.0	1.0
3	1	23.0		0	25	6.84	9.0	3.0	10	1	10.0	4.0	0	5.0	1.0
4	0	47.0		1	33	0.00	7.5	3.0	7	1	3.0	4.0	0	5.0	5.0

Pre-processamento

Separação das features e do target e Scaling das features para treinamento dos modelos

```
from sklearn.preprocessing import StandardScaler

y = train_data['Depression'] # recupera o target
x = train_data.drop(['Depression'], axis=1) # recupera as features

# realiza o escalonamento dos dados
scaler = StandardScaler().set_output(transform='pandas')
scaled_x = scaler.fit_transform(x)
scaled_x
```

Python

	Gender	Age	Working Professional or Student	Professional	CGPA	Sleep Duration	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Work/Academic Pressure	Study/Job Satisfaction
0	1.106796	0.695360	0.497344	-1.667202	-0.486380	1.344493	-1.379133	-0.268958	-0.988861	-1.363057	-0.699592	-0.994217	1.407207	-0.68917
1	-0.903508	-1.161867	0.497344	1.188261	-0.486380	-1.266984	1.065382	0.113698	1.011265	0.193928	0.007813	-0.994217	0.693841	0.02248
2	-0.903508	-0.596624	-2.010679	0.426804	2.386347	-0.483541	-1.379133	-1.034272	1.011265	-0.844062	-1.406997	-0.994217	1.407207	-0.68917
3	-0.903508	-1.484863	0.497344	1.188261	-0.486380	-1.266984	0.250544	-0.651615	1.011265	0.972421	-1.406997	1.005816	1.407207	-1.40083
4	1.106796	-0.838871	0.497344	-1.762385	-0.486380	-0.483541	1.065382	-0.651615	1.011265	0.712923	0.715218	1.005816	-1.446257	-1.40083
...
140695	1.106796	-1.807859	0.497344	0.426804	-0.486380	-0.483541	1.065382	-0.013854	-0.988861	-1.103560	0.715218	1.005816	1.407207	0.73414
140696	1.106796	0.049368	0.497344	-1.286474	-0.486380	0.561050	0.250544	-0.906719	1.011265	-0.065570	1.422623	1.005816	1.407207	0.73414
140697	1.106796	-1.323365	0.497344	0.236440	-0.486380	1.344493	0.250544	-1.289376	-0.988861	-0.584565	0.715218	-0.994217	-0.019525	-1.40083
140698	1.106796	0.695360	0.497344	0.712351	-0.486380	-0.483541	0.250544	1.516773	1.011265	0.972421	-1.406997	-0.994217	1.407207	-0.68917
140699	-0.903508	-1.081118	-2.010679	0.426804	2.472817	-1.266984	-1.379133	-0.524063	1.011265	-1.103560	0.007813	1.005816	0.693841	-1.40083

140700 rows × 14 columns

Treinamento e Modelos

Com os dados tratados e preparados para o treinamento, sera realizado um teste de treinamento em diversos modelos de aprendizagem supervisionada e o que tiver melhor Accuracy Score sera o escolhido para gerar o submission

Assim foi realizado:

- Separação dos dados de treinamento e de testes
- Definição e instanciamento dos modelos
- Treinamento de todos os modelos com os mesmos dados de treino e teste
- Definição do melhor modelo

Treinamento e Modelos

Separação dos dados de treinamento e de testes com train_test_split

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# separacao dos dados de treino e teste
x_train, x_test, y_train, y_test = train_test_split(scaled_x, y, test_size = 0.3, random_state= 42)
```

Python

Treinamento e Modelos

Definição e instanciação dos modelos para aprendizagem supervisionada.
(KNN, SVM, Arvore de Decisão, Regressão Logística e Random Forest)

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

# definição dos modelos a serem testados
models = []
models.append(('KNeighborsClassifier', KNeighborsClassifier()))
models.append(('SVM', SVC(kernel='linear', random_state = 42)))
models.append(('DecisionTreeClassifier', DecisionTreeClassifier(random_state = 42)))
models.append(('LogisticRegression', LogisticRegression(random_state = 42)))
models.append(('RandomForestClassifier', RandomForestClassifier(random_state = 42)))

✓ 0.1s
```

Treinamento e Modelos

Treinamento de todos os modelos com os dados de treino e teste

```
# treinamento e resultado dos modelos
best_model = None
best_accuracy = 0
best_model_name = ''

# percorre todos os modelos
for name, model in models:
    # realiza o treinamento do modelo
    model.fit(train_data, y_train)

    # realiza a predicao usando os dados separados para teste
    y_pred = model.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred) # recupera a acuracia da predicao

    # print do resultado
    msg = "%s: (%f)" % (name, accuracy)
    print(msg)

    # define como melhor resultado caso seja
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_model = model
        best_model_name = name

print()
print('Resultado Final')
print('Melhor Modelo: ' + best_model_name)
print('Acuracia: ' + str(best_accuracy))
```

Python

Treinamento e Modelos

Definição do melhor modelo - Random Forest

```
KNeighborsClassifier: (0.929306)
SVM: (0.935655)
DecisionTreeClassifier: (0.899976)
LogisticRegression: (0.935371)
RandomForestClassifier: (0.935963)
```

Resultado Final

Melhor Modelo: RandomForestClassifier

Acuracia: 0.9359630419331912

Geração do Submission

Por fim, com o modelo com maior precisão para classificação destes dados, foram realizadas as etapas para geração do submission.csv

Os passos realizados foram:

- Escalonamento dos dados de prova
- Predição da classificação do modelo para os dados de prova
- Exportação e visualização do arquivo submission.csv

Geração do Submission

Escalonamento dos dados de teste, predição do melhor modelo e exportação do arquivo submission.csv com as respostas

```
# realiza o escalonamento dos dados de prova
scaler = StandardScaler().set_output(transform='pandas')
scaled_test_x = scaler.fit_transform(test_data)

# realiza a predicao
Y_test = best_model.predict(scaled_test_x)

# salva o resultado no arquivo submission.csv
submission = pd.read_csv('data/sample_submission.csv')
submission['Depression']= Y_test
submission.to_csv('submission.csv', index=False)
submission.head(10)
```

✓ 0.9s

	id	Depression
0	140700	0
1	140701	0
2	140702	0
3	140703	1
4	140704	0
5	140705	0
6	140706	0
7	140707	0
8	140708	0
9	140709	1

Obrigado!

O repositorio com o codigo fonte e arquivos pode ser encontrado em:

**UNIVEM-BCC-BSI/
predicting-depression-...**

Trabalho de avaliação final da matéria de machine learning. Treinamento de um modelo que realiza a predição de indivíduos que...

1 Contributor 0 Issues 0 Stars 0 Forks

UNIVEM-BCC-BSI/predicting-depression-machine-learning: Trabalho de avaliação final da matéria de...

Trabalho de avaliação final da matéria de machine learning. Treinamento de um modelo que realiza a predição de indivíduos que possuem Depressão baseado em dados de entrevistas. - UNIVEM-BCC-BSI/pre...

 GitHub

