RELATÓRIO

Antônio Kauê 558439
José Basílio 552283

INTRODUÇÃO

Neste relatório, apresentamos a análise de dois datasets distintos com o objetivo de explorar técnicas de pré-processamento e análise de dados.

O primeiro dataset usado diz respeito à previsão de Acidente Vascular Cerebral (AVC) em indivíduos com base em características demográficas e de saúde, sendo uma das principais causas de mortalidade no mundo. Este conjunto de dados, composto por informações de pacientes, oferece a oportunidade de investigar fatores como idade, gênero e condições clínicas que influenciam o risco de AVC. A análise busca não apenas compreender esses fatores, mas também estabelecer bases para modelagens preditivas que possam contribuir para intervenções precoces.

O segundo dataset explora o mercado financeiro, especificamente os preços de ações de grandes empresas de tecnologia. Em um cenário onde decisões financeiras rápidas e bem informadas são essenciais, a capacidade de prever tendências de preços de ações pode representar uma vantagem competitiva significativa. Este conjunto de dados traz informações como preço de fechamento e volume de negociações, permitindo a aplicação de métodos estatísticos para entender padrões e fazer previsões.

Nesse documento é mostrado o processo de análise e modelagem realizado em ambos os datasets. Em cada uma das seções, são apresentados como os códigos foram implementados, explicando cada métrica, como foi cada dataset foi manipulado para chegar no resultados obtidos.

OBSERVAÇÕES

Durante o processo de criação do projeto, algumas dificuldades atrapalharam bastante, uma delas foi a de analisar os Datasets, onde a gente teve bastante dificuldade de interpretar os dados que os nossos Datasets fornecia, juntamente com esse problema, a criação dos códigos foram bastantes complexos. A gente não sabe muito bem programar e juntando a dificuldade de analisar o Dataset e com a dificuldade de programar, se tornou um projeto bastante complicado de se fazer. Diante disso, fizemos o máximo que estava ao nosso alcance.

RESULTADOS E DISCUSSÃO

QUESTÃO 1

O objetivo da questão é realizar uma manipulação em um dataset. utiliza-se a biblioteca panda e realiza o pré-processamento nele.

Código

1. O código começa importando a biblioteca **pandas** para ler o arquivo CSV.
2. O próximo passo é a verificação da existência de valores nulos **isnull().sum().any(),** caso tenhas valores faltando, utiliza-se o **dropna()** para removê-los.
3. Uma lista de colunas relevantes para a análise é definida e usada para criar um novo dataset, onde é exibido mostrando as 5 primeiras colunas relevantes.
4. A função **class_distribuition** calcula a distribuição de classes da coluna **stroke**, que representa a ocorrência de AVC, e imprime os valores em percentual.
5. Renomeação de valores: Caso a coluna **stroke** seja do tipo **object,** seus valores são mapeados para 0 (não teve AVC) e 1 (teve AVC).
6. Pré-processamento: para evitar a multicolinearidade, as variáveis categóricas são transformadas em variáveis binárias.
7. salvamento do dataset: agora o dataset pré-processado é salvo como: **stroke_prediction_dataset_ajustado.csv**

Resultados

```
DataFrame Final:
   gender   age  hypertension  ...   bmi   smoking_status  stroke
0    Male  67.0             0  ...  36.6  formerly smoked       1
2    Male  80.0             0  ...  32.5     never smoked       1
3  Female  49.0             0  ...  34.4           smokes       1
4  Female  79.0             1  ...  24.0     never smoked       1
5    Male  81.0             0  ...  29.0  formerly smoked       1

[5 rows x 8 columns]

Distribuição de classes (em %):
stroke
0    95.742514
1     4.257486
Name: proportion, dtype: float64

Dataset ajustado salvo como: stroke_prediction_dataset_ajustado.csv
```

O novo dataset slvo como **stroke_prediction_dataset_ajustado.csv** é mostrado mostrando as 5 primeiras colunas mais relevantes após o pré-processamento; A distribuição de classe em porcentagem da coluna **stroke** mostra que mostra que 95,74% dos casos não apresentaram AVC (classe 0) e apenas 4,26% tiveram AVC (classe 1).

Problemas

O desequilíbrio de classes foi um problema encontrado, pois esse desequilíbrio pode afetar o modelo de aprendizagem de máquina.

QUESTÃO 2

O objetivo da questão é realizar uma classificação utilizando KNN implementado de forma manual.

Código

1. O código começa carregando o dataset ajustado que foi salvo anteriormente, utilizando a biblioteca **pandas** para ler o arquivo CSV.
2. Após isso, o dataset é dividido em variáveis independentes (**X**), que são todas as colunas, exceto a coluna **stroke**, e a variável alvo (**y**), que é a coluna stroke, indicando se o paciente teve ou não um AVC.
3. O conjunto de dados é dividido em treino e teste com a função **train_test_split** da **sklearn**, utilizando 80% dos dados para treino e 20% para teste. A normalização das variáveis independentes é realizada usando **StandardScaler**, garantindo que os dados de entrada tenham uma média de 0 e desvio padrão de 1, o que é importante para o desempenho de algoritmos baseados em distância, como o **KNN**.
4. A função knn implementa o algoritmo KNN (K-Nearest Neighbors) de forma simplificada, considerando diferentes métricas de distância para calcular a proximidade entre os pontos. As métricas possíveis são: **Mahalanobis, Chebyshev, Manhattan e Euclidiana.** A função também realiza a previsão da classe para cada ponto de teste (**`X_test`**) com base nas k amostras mais próximas (usando a métrica escolhida).
5. O código desenvolvido visa testar o modelo com diferentes métricas de distância (**Mahalanobis, Chebyshev, Manhattan e Euclidiana**) e calcula a acurácia da classificação, comparando as previsões (**y_pred**) com os valores reais do conjunto de teste (**y_test**).

Resultados

```
Resultados:
Acurácia com métrica mahalanobis: 0.96
Acurácia com métrica chebyshev: 0.96
Acurácia com métrica manhattan: 0.96
Acurácia com métrica euclidean: 0.96
```

O resultado apresenta as acurácias para cada uma das métricas de distância utilizadas no KNN. Todos os valores de acurácia são 0.96 (ou 96%) para todas as métricas testadas. Isso significa que o modelo teve uma excelente capacidade de acertar as previsões para as quatro métricas de distância. A alta acurácia indica que o modelo está fazendo uma boa previsão da ocorrência de AVC, independentemente da métrica usada.

Questão 3

O objetivo da questão é buscar a melhor parametrização do knn implementado na segunda questão. é utilizado dois tipos de normalização: logarítmica e média zero e variância unitária.

Código

1. O código começa utilizando o **pandas** e carrega o dataset previamente salvo **stroke_prediction_dataset_ajustado.csv** e separa as variáveis.
2. Implementação do KNN: A função knn vai usar a métrica de distância euclidiana e ela vai calcular a distância entre o ponto de teste e o ponto de treino **x_test, x_train.** Em seguida Seleciona os k vizinhos mais próximos usando **np.argsort.**
3. O conjunto de dados são divididos em 80% para treino e 20% para teste, para isso utilizamos a função **train_test_split.**
4. utilizamos a normalização logarítmica nos dados para reduzir influência de valores extremos e tornar mais simétricos. Média Zero e Variância Unitária: Utiliza o**StandardScaler** para escalar os dados de forma que cada variável tenha média 0 e desvio padrão 1, adequado para algoritmos sensíveis à escala.
5. Em seguida é printado a acurácia para as duas normalizações.

Resultados

```
Acurácia com normalização logarítmica: 0.96
Acurácia com média zero e variância unitária: 0.96
```

As mesmas técnicas de normalização resultaram em valores semelhantes, isso quer dizer que para esse dataset e esse modelo, as duas técnicas de normalização são eficazes. A escolha de uma técnica de normalização pode variar de acordo com o dataset com diferentes distribuição de dados. .

QUESTÃO 4

O objetivo da questão é buscar saber a melhor parametrização do knn implementado na questão anterior (questão 3).

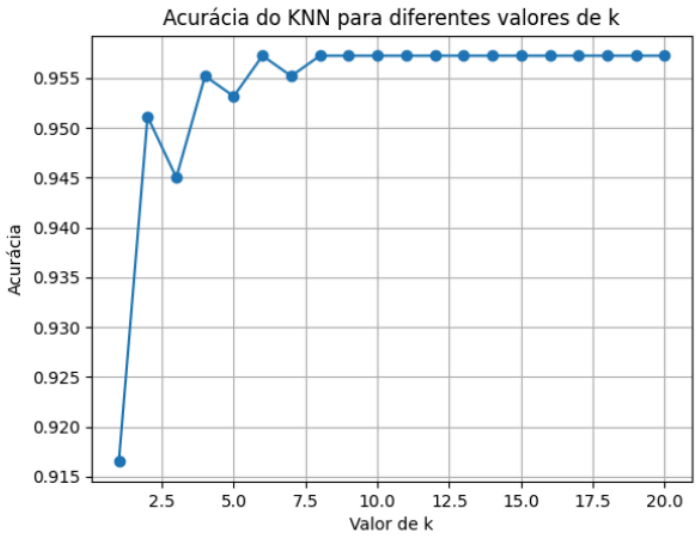
Código

1. É utilizado o dataset ajustado e previamente salvo, separando as variáveis independentes (**x**) e a variável alvo (**y**).
2. Função KNN Otimizada: A função knn usa a função cdist da biblioteca scipy para calcular as distâncias euclidianas entre os pontos de teste (**X_test**) e os pontos de treino (**X_train**) de forma vetorizada, o que melhora a eficiência. Para cada ponto de teste, identifica os índices dos **k** vizinhos mais próximos (**k_nearest_indices)** e as respectivas classes (**k_nearest_labels),** no fim a classe final de cada ponto é determinada pela votação majoritária entre os **k** vizinhos.
3. Os dados são divididos em treino (80%) e teste (20%) com **train_test_split**. Em seguida, os dados são escalados para média zero e variância unitária usando **StandardScaler**, o que é importante para garantir que o cálculo da distância seja consistente entre as variáveis.
4. Após isso, o código avalia o modelo para diferentes valores de **k** variando de 1 a 20. e para cada valor de **k** a acurácia é calculada como a proporção de previsões corretas.
5. Um gráfico gerado relaciona os valores de **k** com a acurácia obtida, ajudando a visualizar o comportamento do modelo à medida que o **k** varia.

Resultados

```
O melhor valor de k é: 6 com acurácia de 0.96
```

O resultado mostrou que o valor de **k** é 6, alcançando uma acurácia de 96%. O valor de **k** oferece o melhor equilíbrio entre sensibilidade e generalização para este dataset.



O gráfico ajuda a entender como a escolha de **k** afeta o desempenho do modelo, pois em geral, valores muito baixos de **k** (como 1) podem levar a um modelo sensível ao ruído, por outro lado, valores muito altos de **k** podem suavizar demais as decisões, levando a um modelo menos sensível a padrões locais.

Questão 5

O objetivo da questão é utilizar outro dataset (regressão) e realizar o pré-processamento, em seguida fazer uma análise para verificar qual atributo será alvo para a regressão no dataset e qual atributo mais relevante para realizar a regressão do alvo escolhido. De forma mais específica, tem o objetivo de analisar a correlação entre variáveis financeiras e o preço de fechamento utilizando um gráfico de calor para a correlação.

Código

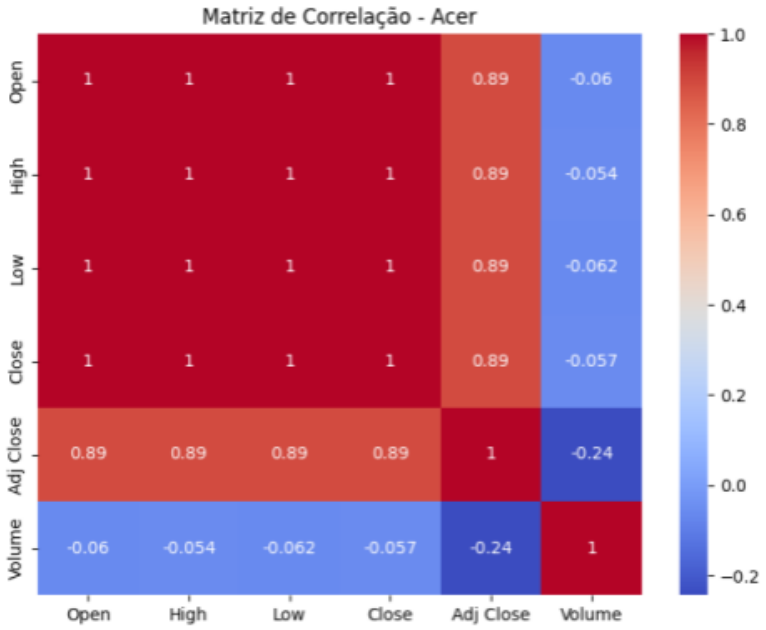
1. O código começa carregando o novo dataset onde contém informações de ações da empresa Acer no período de 2000 a 2024, e utiliza o **pandas, seaborn e matplotlib.**

2. A conversão de dados é feita o que permite uma análise baseada em séries temporais, utilizando o **pd.to_datetime**
3. O **df_numerico = df.select_dtypes(include=['float64', 'int64'])** para fazer uma seleção de apenas colunas numéricas pois o cálculo de correlação só pode ser feito através de colunas numéricas.
4. A correlação entre todas as colunas numéricas e a coluna **Close** (Preço de Fechamento) é calculada usando o método **Corr(),** após isso os valores são ordenados de forma decrescente.
5. Plotagem: utilizamos a biblioteca Seaborn para fazer um gráfico de calor da matriz de correlação onde as cores representam o grau de correlação.

Resultados

```
Correlação com o Preço de Fechamento (Close):
 Close      1.000000
High        0.999626
Low         0.999619
Open        0.999143
Adj Close   0.891128
Volume     -0.056976
Name: Close, dtype: float64
```

Essas foram as colunas mais correlacionadas com o preço de fechamento. As variáveis **high, low e open** tem uma correlação quase perfeita com o fechamento que é o **Close**. A correlação negativa indica que o volume de negociação não está relacionado com o preço de fechamento.



O gráfico vai fornecer uma visão geral entre todas as variáveis

Questão 6

O objetivo da questão é criar um modelo de regressão linear para prever o preço de fechamento do dataset da questão 5 utilizando algumas métricas.

Código

1. O código vai começar carregando o dataset utilizando na questão 5 com os dados financeiros da Acer. há uma conversão da coluna **date** para **datetime.**
2. utiliza-se o mesmo processo da quinta questão onde seleciona apenas colunas numéricas, pois a análise de regressão linear exige que os dados sejam numéricos.
3. Com base na correlação obtida anteriormente, foi escolhia a variável **high** como variável independente(X) e o preço de fechamento **close** vá ser a variável dependente (Y)
4. Um modelo de regressão linear simples é ajustado usando **LinearRegression** do scikit-learn. a partir disso o modelo é treinado para encontrar a relação entre a variável **high** e o fechamento **close.**
5. utiliza-se o **RSS** Soma dos quadrados dos resíduos (diferenças entre valores reais e preditos). **MSE:** Média dos quadrados dos erros. **RMSE:**Raiz quadrada do MSE, para interpretar o erro na mesma escala dos dados. **R^2:** Mede o quanto da variação em **Close** é explicada por **High.**

Resultados

RSS: número baixo o que indica poucos resíduos
MSE/RMSE: Valores baixos, o que vai confirmar que o erro médio de predição é pequeno.
R^2: Um valor muito próximo de 1 (como 0.99 ou mais), indicando uma excelente explicação da variável Close pela variável High. indica que os preços variam de forma previsível pois a relação entre o preço mais alto do dia high e o preço de fechamento é quase linear.

CONCLUSÕES

O trabalho desenvolvido mostrou a importância de técnicas de pré-processamento, normalização e seleção de atributos na melhoria de modelos preditivos. Para o dataset de AVC, o modelo KNN demonstrou alta acurácia independente das métricas de distância ou normalização utilizada. Já para o dataset da Acer, a regressão linear confirmou a forte relação entre o preço de fechamento e outras variáveis, permitindo previsões confiáveis.

PRÓXIMOS PASSOS

1. Testar novos algoritmos para avaliar o desempenho de outros modelos
2. Para melhorar a previsão de casos positivos no dataset de AVC, utilizar outras técnicas de balanceamento.
3. Fazer uma análise outra variáveis, uma melhor análise foi feita em cima da variável **high**, mas poderia explorar novas variáveis.