UDACITY

< Back to Machine Learning Engineer Nanodegree

Creating Customer Segments

REVISÃO

HISTORY

Meets Specifications

O seu trabalho está muito bom. Parabéns!
Continue assim para manter sua trajetória excepcional.
😄 😄 😄

Boa sorte em seus próximos projetos!

Se quiser me adicionar no Linkedin (Rafael Buck) fique à vontade.

Exploração dos dados

Três amostras diferentes dos dados são escolhidas, e o que elas representam é proposto com base na descrição estatística dos dados.

Top! Excelente descrição dos exemplos selecionados. Ficou bem legal.

Sugestão: a parte de traduzir os dados em exemplo concretos para um público mais business é uma das mais importantes na área de *data science* e *machine learning*. Recomendo sempre treinar isso 😄

A pontuação do atributo removido foi corretamente calculada. A resposta justifica se o atributo removido é relevante.

Excelente, devidamente implementado. Uma coisa legal de se observar é que os atributos de menor pontuação são mais relevantes para o modelo já que não pode ser previstos. Eles são menos afetados pelo efeito de colinearidade. Já os atributos com maiores pontuações são facilmente previstos pelos outros atributos, portanto não estão trazendo nenhuma informação nova para a análise.

Sugestão: aqui um artigo legal que elenca a questão da multi-colinearidade. E segue outra referência sobre técnicas de remoção de parâmetros antes de realizar as predições
Abaixo um plus, um exemplo de como você fazer o cálculo para todas variáveis:

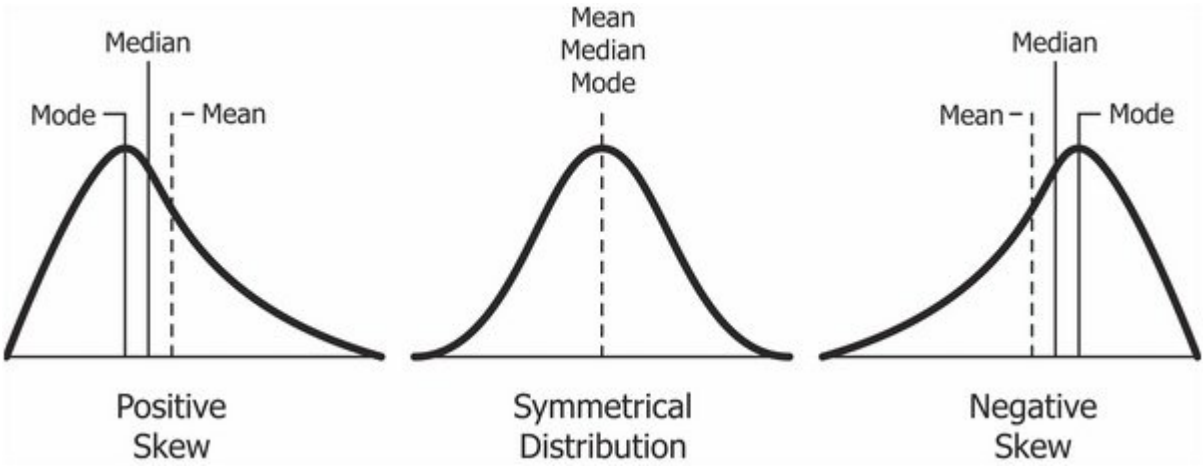
```
# FORFUN: Testando todos
for column in data.columns.values:
    # Fazer uma cópia do DataFrame utilizando a função 'drop' para soltar o atributo dado
    target_array = data[column]
    new_data = data.drop(column, axis = 1)

    # Dividir os dados em conjuntos de treinamento e teste utilizando o atributo dado como o alvo
    X_train, X_test, y_train, y_test = train_test_split(new_data,
                                                        target_array,
                                                        test_size=0.25,
                                                        random_state=50)

    # Criar um árvore de decisão regressora e ajustá-la ao conjunto de treinamento
    regressor = DecisionTreeRegressor(random_state=50)
    regressor = regressor.fit (X_train,
                               y_train)
    y_pred = regressor.predict(X_test)

    # Reportar a pontuação da previsão utilizando o conjunto de teste
    score = regressor.score(X_test,
                            y_test)
    if (score < 0):
        print("Score (R^2) negativo para atributo {} (modelo falhou em ajustar os dados)".format(c
olumn))
    else:
        print("Score (R^2) para atributo {} é de {}".format(column, score))
```

```
Score (R^2) negativo para atributo Fresh (modelo falhou em ajustar os dados)
Score (R^2) para atributo Milk é de 0.15610348561723175
Score (R^2) para atributo Grocery é de 0.7127493539819663
Score (R^2) negativo para atributo Frozen (modelo falhou em ajustar os dados)
Score (R^2) para atributo Detergents_Paper é de 0.7780991929451163
Score (R^2) negativo para atributo Delicatessen (modelo falhou em ajustar os dados)
```

Atributos correlacionados são corretamente identificados e comparados com o atributo previsto. A distribuição dos dados para esses atributos é discutida.

Muito bons os resultados. O legal deles é que você consegue verificar a relação encontrada anteriormente. O `Freso` não tem relação com nada, enquanto `Grocery` e `Detergents_Paper` são praticamente redundantes. Outra coisa bacana é notar que os dados estão distorcidos positivamente, mais concentrados na origem, por isso o pré-processamento do item seguinte é feito nos dados.

Sugestão: Abaixo um exemplo dos tipos de distribuição que é importante observamos nessa etapa da análise:



E aqui um artigo muito bacana de como os outliers afetam as distribuições.

Pré-processamento dos dados

Os valores aberrantes extremos são identificados, e discute-se se eles deveriam ser removidos. A decisão de remover quaisquer dados é corretamente justificada.

Ficou muito boa sua análise de como identificar os outliers.

Sugestão: segue um artigo sobre como lidar com lidar com outliers. E esse artigo também discute sobre remover ou não outliers.

O código de dimensionamento de atributos, tanto para os dados como para as amostras, foi corretamente implementado.

Excelente! Uma alternativa seria o uso do Box-Cox ou até do `preprocessing.scale` do Sklearn 😄

Sugestão: aqui um link bem bacana de várias técnicas de como fazer a transformação normal de uma distribuição. Abaixo um trecho de código, caso queira implementar 😄

```
# FORFUN: Exemplo usando Box-Cox (se for usar, tem que aplicar em um atributo por vez)
from scipy.stats import boxcox
import matplotlib.pyplot as plt

for column in data.columns.values:
    boxcox_transformed_data = boxcox(data[column])
    plt.figure() #this creates a new figure on which your plot will appear
    plt.title(column)
    sns.distplot(boxcox_transformed_data[0])
```

Transformação de atributos

A variância explicada total para duas e quatro dimensões dos dados do PCA é corretamente relatada. As primeiras quatro dimensões são interpretadas como uma representação dos gastos do cliente com justificativa.

Perfeito, as variâncias explicadas estão corretíssimas. Veja que com apenas 2 componentes principais já é possível fazer uma clusterização bastante representativa. Também é importante notar a relevância do valor absoluto de cada categoria naquela dimensão.

O código do PCA foi corretamente implementado e aplicado, tanto para os dados dimensionados como para as amostras dimensionadas, no caso bidimensional.

Perfeito, devidamente implementado.

Sugestão: segue um artigo que ilustra bem um exemplo visual do PCA. Esse artigo aqui também discute como analisar cada dimensão.

Clustering

Os algoritmos GMM e k-means são comparados em detalhes. A escolha do aluno é justificada com base nas características do algoritmo e dos dados.

Excelente explicação do K-means e do GMM e a justificativa da escolha do algoritmo também ficou muito boa.

Sugestão: segue anexado o link de um artigo que discute exatamente isso, bastante interessante. Caso queira testar o GMM, eu fiz a implementação no meu trabalho: https://github.com/rafaelmartinsbuck/machine-learning/blob/master/creating-customer-segments/customer_segments_PT.ipynb se quiser se divertir segue como referência 😄

Amostras dos dados são corretamente relacionadas aos segmentos da clientela, e o grupo a que pertence cada ponto da amostra é discutido.

Aqui é muito bacana. É onde retomamos uma linguagem mais business, explicando o que o respectivo cluster representa em termos mais práticos.

Sugestão: novamente, é uma oportunidade muito boa de ir treinando isso, que consiste em uma das habilidades mais importantes na área de *data science* e *machine learning* 😄

Diversas pontuações são corretamente relatadas, e o número ótimo de grupos é escolhido com base na melhor. A visualização escolhida mostra o número ótimo de grupos baseado no algoritmo de clustering escolhido.

Perfeito, grupos propostos da forma correta!

Os grupos representados por cada segmento da clientela são propostos com base na descrição estatística do conjunto de dados. Os códigos de transformação e dimensionamento inversos foi corretamente implementado e aplicado para o centro dos grupos.

A análise dos clientes está boa. Parabéns!

Conclusão

O aluno identifica corretamente como um teste A/B pode ser feito com a clientela após uma mudança no serviço de distribuição.

Análise excelente. Parabéns pelo trabalho e espero que tenha se divertido! 😄

O aluno discute e justifica como os dados de clustering podem ser usados em um modelo de aprendizagem supervisionada para fazer novas estimativas.

Excelente a resposta de como usar os dados para aprendizagem supervisionada. Com essa segmentação é possível entender melhor os clientes e oferecer melhores estratégias de abordagem a cada segmento.

Os segmentos da clientela e os dados em `Channel` são comparados. Os segmentos identificados pelos dados de `Channel` são discutidos, inclusive se essa representação é consistente com resultados anteriores.

Top! Muito boa a análise de como executar um teste A/B para esse problema.

Sugestão: é importante observar que se for testado somente um cluster, pode obter um resultado totalmente diferente no outro. Em um teste A/B seria importante entender cada cluster como um tipo de cliente distinto, sem misturar, para assim ter um conhecimento maior dos seus clientes. Segue um artigo interessante do blog do Netflix sobre o tema. Segue também uma referência de como conduzir um teste A/B na prática, o que levar em conta na hora de planejá-lo e executá-lo 😄

BAIXAR PROJETO

RETORNAR