Projeto de Análise e Previsão de Índice de Desenvolvimento Humano (IDH)

O projeto consiste em analisar a possível relação entre o IDH e outros índices que possam ter influência sobre ele, com isso podendo prever com precisão satisfatória um possível aumento do IDH baseado nesses índices.

Índices Utilizados

1- Produto Interno Bruto (PIB)
2- Índice de Liberdade Econômica (ILE)
3- Índice de Percepção da Corrupção (IPC)

Extração dos Dados

Os dados referem-se ao ano de 2019 e foram extraídos de diferentes sites, conforme abaixo:

1- PIB:
https://pt.wikipedia.org/wiki/Lista_de_pa%C3%ADses_por_PIB_nominal

2- ILE:
https://www.fraserinstitute.org/economic-freedom/map?geozone=world&page=map&year=2019

3- IPC:
https://www.transparency.org/en/cpi/2019

4- IDH:
https://www.cnnbrasil.com.br/internacional/veja-o-ranking-completo-de-todos-os-paises-por-idh/

Tratamento dos Dados

Como os dados foram adquiridos de fontes diferentes, é natural que eles tivessem formatos diferentes, então foram necessários procedimentos de tratamento para que fosse possível submetê-los aos experimentos e estes pudessem ter uma maior precisão e performance e, após extração, os dados foram inseridos no Excel para serem pré-tratados conforme indicado abaixo:

Estruturação: Colocar os dados em uma estrutura que fosse possível relacioná-los.

Consistência: Nomes em diferentes idiomas para o mesmo país. Foi utilizado o tradutor para que todos os países tivessem seus nomes em conformidade com português-BR.

Conformidade: Países que não tiveram dados de um ou mais índices divulgados foram eliminados do processo, garantindo assim que todo nosso estudo e pesquisa contasse com dados reais e precisos.

Após tratamento adequado os dados foram exportados em formato “.csv” para que pudessem ser utilizados para experimentos no Python.

Experimentos

Correlação

Com a correlação, buscou-se verificar a correlação entre as variáveis independentes (PIB, ILE, IPC) e a variável dependente (IDH).
O resultado mostra a inexistência da correlação do PIB, correlação positiva moderada do ILE e correlação positiva forte do IPC.

```
[6]: #Verificando a correlação entre PIB e IDH
     corrPIB = np.corrcoef(X[:,0], y)
     corrPIB

[6]: array([[1.        , 0.20176387],
            [0.20176387, 1.        ]])

[7]: #Analisando a correlação entre ILE e IDH
     corrILE = np.corrcoef(X[:,1], y)
     corrILE

[7]: array([[1.       , 0.6800175],
            [0.6800175, 1.       ]])

[8]: #Analisando a correlação entre IPC e IDH
     corrIPC = np.corrcoef(X[:,2], y)
     corrIPC

[8]: array([[1.        , 0.76337469],
            [0.76337469, 1.        ]])
```

Devido ao fato da correlação do PIB ter se mostrado inexistente, ao analisar mais a fundo, observou-se que países como os Estados Unidos e China, ambos responsáveis pela maior parcela do PIB mundial, com 24% e 17% respectivamente, não aparecem nem entre os 5 primeiros colocados no índice IDH. Esse fator corrobora com o resultado de inexistência dessa correlação, pois ao contrário disso, os países com maior PIB estariam entre os primeiros colocados do ranking de IDH. Com isso podemos retirar o atributo PIB do estudo.

Primeiros e últimos colocados mundiais em IDH

```
[9]: #Ordenando por maior IDH
     base.sort_values(['IDH'], ascending = False)
```

[9]:

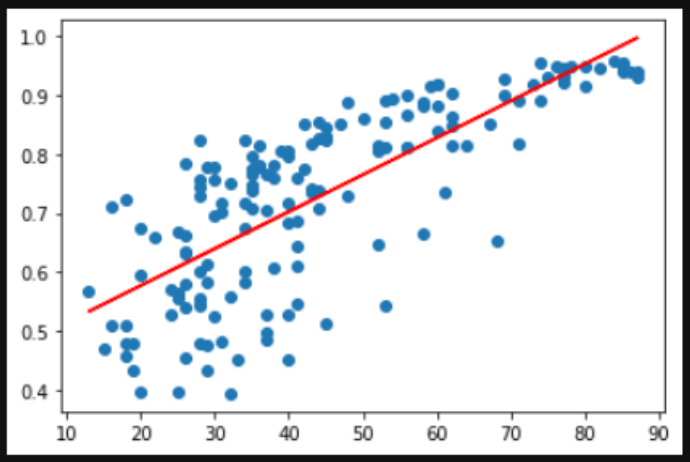
	País	PIB	ILE	IPC	IDH
113	Noruega	0.005256	7.72	84	0.957
78	Irlanda	0.004634	8.21	74	0.955
141	Suíça	0.008570	8.48	85	0.955
70	Hong Kong	0.004391	8.91	76	0.949
79	Islândia	0.000314	7.90	78	0.949
...	...	...	...	...	...
98	Mali	0.000020	5.83	29	0.434
25	Burundi	0.000041	5.65	19	0.433
33	Chade	0.000133	5.60	20	0.398
126	República Centro-Africana	0.000002	5.36	25	0.397
111	Níger	0.000112	5.97	32	0.394

Regressão Linear Simples

A regressão Linear demonstra a evolução positiva entre as variáveis ILE e IPC com o IDH, ou seja, quanto maior for a pontuação desses índices, maior tende a ser o Índice de Desenvolvimento Humano de um país. Embora podemos realizar previsões apenas com uma variável, o objetivo com a regressão é apenas visualizar a dispersão dos dados juntamente com a linha de melhor ajuste.

Dispersão da IPC

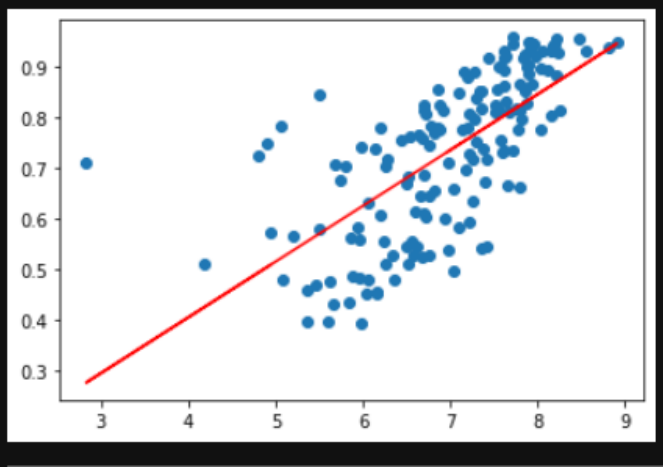
```
[9]: #Visualisando gráfico de dispersão de X
     plt.scatter(X, y)
     plt.plot(X, modeloX.predict(X), color = 'red')

[9]: [<matplotlib.lines.Line2D at 0x260bade63d0>]
```




```
[10]: #Previsão do IDH utilizando a variável ICP
      modeloX.predict([[50]])

[10]: array([0.7647553])
```

Dispersão da ILE

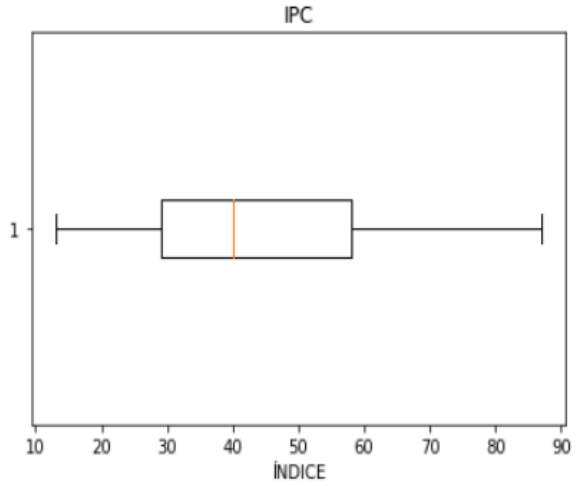
```
[15]: #Visualizando gráfico de dispersão de Z
      plt.scatter(Z, y)
      plt.plot(Z, modeloZ.predict(Z), color = 'red')

[15]: [<matplotlib.lines.Line2D at 0x260bcee0160>]
```

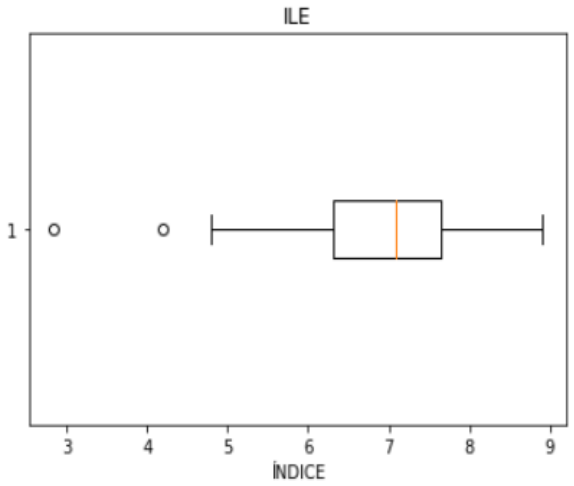


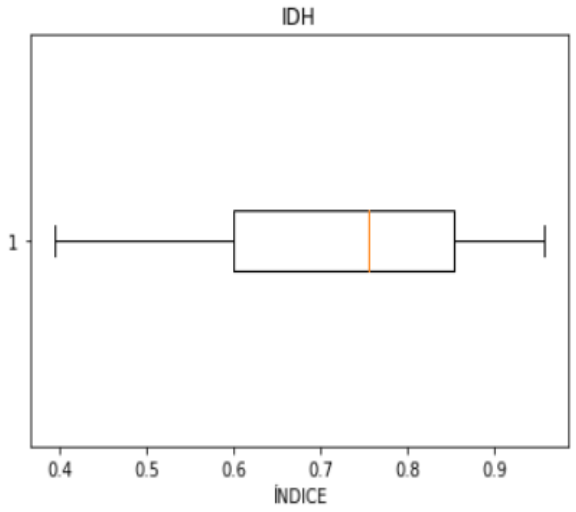
No gráfico de dispersão de ILE, podemos notar um provável outlier mais a esquerda da imagem, que foi confirmado posteriormente no gráfico boxplot e na captura desse outlier através de um método criado utilizando z-score.

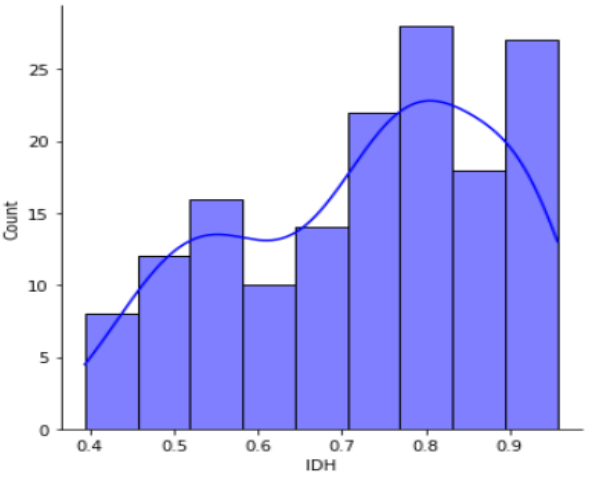
Gráficos

Aqui seguimos analisando a distribuição dos dados e possíveis outliers de forma individual para cada atributo e a classe.





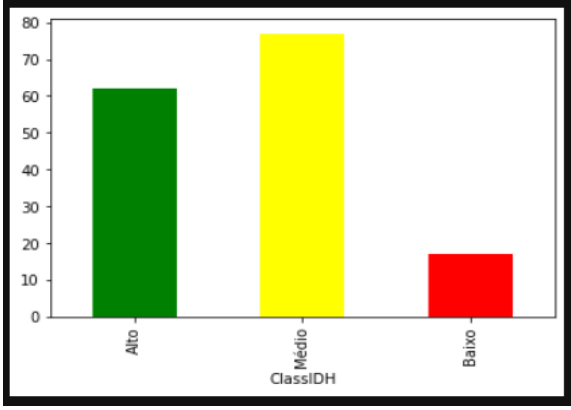





```
ClassIDH
Alto     62
Médio    77
Baixo    17
dtype: int64
```

```
#Gráfico de barras da classe
classe.plot.bar(color = ['green', 'yellow', 'red'])
```

```
<AxesSubplot:xlabel='ClassIDH'>
```



Analisando os gráficos individualmente, podemos extrair algumas informações, como por exemplo, a presença de possíveis outliers em ILE, o diferencial de escalonamento entre os atributos ILE e IPC e o desbalanceamento das Classes.
A princípio, a única correção a ser efetuada foi a de remoção de outliers, pois como podemos ver a seguir, considerando 3 desvios padrão, foi identificado apenas um outlier e este não pertence a classe que possui poucas amostras, fazendo com que essa remoção não prejudique o modelo.

Identificação de outliers em ILE

```
#Capturando outliers no índice ILE
outliers = []

def find_outliers(ile):
    corte_dpadrao = 3
    media = np.mean(ile)
    dpadrao = np.std(ile)

    for dado in ile:
        z_score = (dado - media)/dpadrao
        if np.abs(z_score) >= corte_dpadrao:
            outliers.append(dado)

    return outliers
```

```
#Visualizando outliers em ILE
outliers = find_outliers(ile)
outliers
```

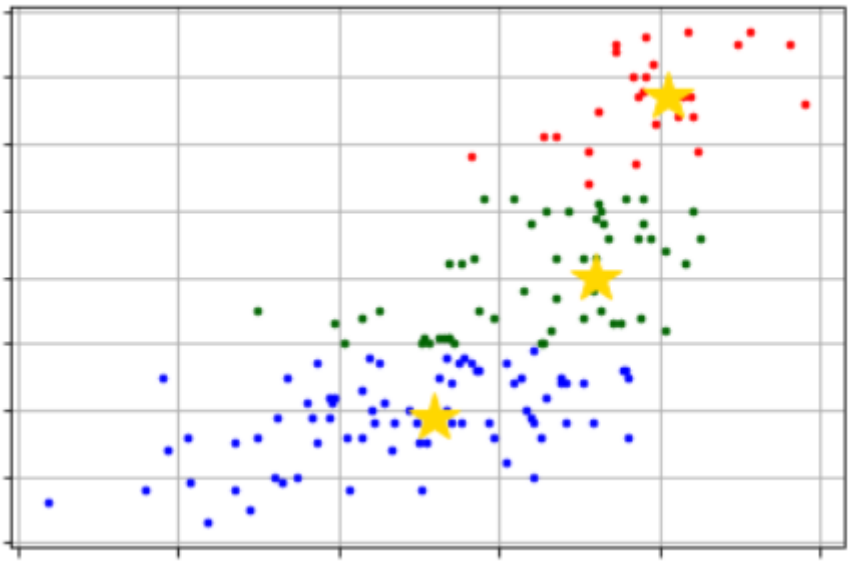
```
[2.83]
```

Agrupamento K-means e K-medoids

Para que pudéssemos utilizar técnicas de agrupamento, foi necessário a transformação das pontuações do IDH para classe hierárquica, sendo assim essa classe foi dividida em 3 grupos.

Com o K-means obteve-se uma taxa de acerto em torno de 30% no agrupamento da classe, já com K-medoids essa taxa foi em torno de 40%.

IDH	Pontuação
Alto	0,8 até 1
Médio	0,5 até 0,799
Baixo	0 até 0,499



Agrupamento com K-medoids

A diferença entre K-means e K-medoids deve-se ao fato de que no K-medoids o centro de inicialização dos clusters são definidos manualmente, podendo assim serem ajustados para tentar obter melhores resultados.

Classificação Random Forest

Por fim, foram aplicadas técnicas de classificação de Machine Learning para avaliar o quão preciso seria um modelo de previsão utilizando tais técnicas.

Configurando o algoritmo com 80% dos dados para treino, 20% para testes e um total de 100 treinamentos, conseguiu-se uma taxa de acerto nos testes em torno de 67%.

```
#Dividindo dados entre treino e teste definindo 30% para teste
X_treino, X_teste, y_treino, y_teste = train_test_split(X,
                                                        y,
                                                        test_size = 0.2,
                                                        random_state = 1)
```

```
#Aplicando algoritmo de ávores aleatórias Random Forest para o treinamento do modelo
forest = RandomForestClassifier(n_estimators = 100)
forest.fit(X_treino, y_treino)
```

```
RandomForestClassifier()
```

```
#Criando a Matriz de Confusão para verificar a taxa de acerto
X = forest.predict(X_teste)
confusion = confusion_matrix(y_teste, X)
confusion
```

```
array([[9, 0, 4],
       [0, 3, 1],
       [1, 4, 9]], dtype=int64)
```

```
#Verificando a performance de acerto da classificação
taxa_acerto = accuracy_score(y_teste, X)
taxa_acerto
```

```
0.6774193548387096
```

Classificação Neural Networks

Com a técnica de Redes Neurais, configuramos os dados de teste basicamente da mesma forma que Árvores de Decisão, com 80% dos dados para treino, 20% para testes e um total de 500 epochs, obtendo assim uma taxa final de acerto nos testes em torno de 77%.

```
#Dividindo dados entre treino e teste definindo 30% para teste
X_treino, X_teste, y_treino, y_teste = train_test_split(normalizados,
                                                        classe_dummy,
                                                        test_size = 0.2,
                                                        random_state = 1)
```

```
#Criação da estrutura da rede neural com a classe Sequential
modelo = Sequential()
#Primeira camada oculta e camada de entrada
modelo.add(Dense(units = 3, input_dim = 2))
#Segunda camada oculta
modelo.add(Dense(units = 3))
#Função de ativação Softmax para classificação de mais de duas
#classes (é gerada uma probablidade em cada neurônio)
modelo.add(Dense(units = 3, activation = 'softmax'))
```

```
#Sumário da estrutura
modelo.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 3)                 9

 dense_1 (Dense)             (None, 3)                 12

 dense_2 (Dense)             (None, 3)                 12

=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0
_________________________________________________________________
```

```
#Criando a Matriz de Confusão para verificar a taxa de acerto
confusao = confusion_matrix(y_teste_matriz, y_previsao_matriz)
confusao
```

```
array([[11,  0,  2],
       [ 0,  0,  4],
       [ 1,  0, 13]], dtype=int64)
```

```
#Verificando a taxa de acerto das previsoes
taxa_acerto = accuracy_score(y_teste_matriz, y_previsao_matriz)
taxa_acerto
```

```
0.7741935483870968
```

Melhoria

Ao final do processo, mesmo com uma taxa satisfatória no acerto das previsões, decidi testar mais um experimento para tentar melhorar essa taxa. A idéia foi que, como estava trabalhando com dados em diferentes escalas, talvez realizando a normalização destas, fosse possível um impacto perceptível para o modelo de previsão e, surpreendentemente, apesar de estarmos lidando com classes desbalanceadas, o resultado da normalização apresentou um resultado perceptível na taxa de acertos na previsão do modelo.

```
#Criando a Matriz de Confusão para verificar a taxa de acerto
confusao = confusion_matrix(y_teste_matriz, y_previsao_matriz)
confusao
```

```
array([[12,  0,  1],
       [ 0,  1,  3],
       [ 0,  1, 13]], dtype=int64)
```

```
#Verificando a taxa de acerto das previsoes
taxa_acerto = accuracy_score(y_teste_matriz, y_previsao_matriz)
taxa_acerto
```

```
0.8387096774193549
```

Conclusão

Ao final do projeto percebe-se que, apesar do desbalanceamento das classes no sentido de ter poucas amostras de países com IDH baixo, o modelo obteve uma taxa de acerto satisfatória nas previsões dos resultados.

Em relação às classes desbalanceadas, fica impossibilitado a correção por meio de nivelamento por corte, em vista de que os dados já possuem poucas amostras. Já na correção por meio de inserção de mais dados na classe de menor quantidade, poderia ocorrer super ajuste do resultado pelo fato de que seriam muitos dados a serem inferidos nessa classe.

Por fim podemos concluir que o Índice de Liberdade Econômica e o Índice de Percepção da Corrupção parecem afetar de forma significativa o Desenvolvimento Humano de uma população e, com os modelos de previsão utilizados neste estudo, em especial Redes Neurais, pode-se almejar um aumento do IDH com base na melhoria do ILE e principalmente do IPC, podendo obter assim uma taxa de sucesso de previsões do alvo principal em torno de 80%.