Proyecto Inteligencia Artificial

FLIGHT
PROBABILITY

Artificial Intelligence

Universidad Industrial de Santander

Juan Diego Esteban Parra - 2190063

Esteban David Florez Tolosa - 2191940

Jesus Darío Villamizar Bohada - 2191960

Planteamiento del Problema

Existe una problemática recurrente en la mayoría de compañías aéreas la cual arruina la experiencia de muchos usuarios, el cancelación de vuelos es una problemática que puede llegar a tener un impacto negativo en los viajeros y que muchas aerolíneas pierdan clientes.

Para abordar este desafío, mi equipo ha decidido crear una inteligencia artificial que pueda predecir la cancelación de vuelos. Para ello, vamos a utilizar un dataset de viajes en avión dentro de Brasil que incluye un total de 10048575 filas por 15 columnas. Este dataset que escogimos contiene información sobre las aerolíneas, las ciudades de origen, los motivos de retraso y otros factores relevantes.

```
a.shape
#codido = (a.Codigo.Justificativa = ACTIVADA )
```

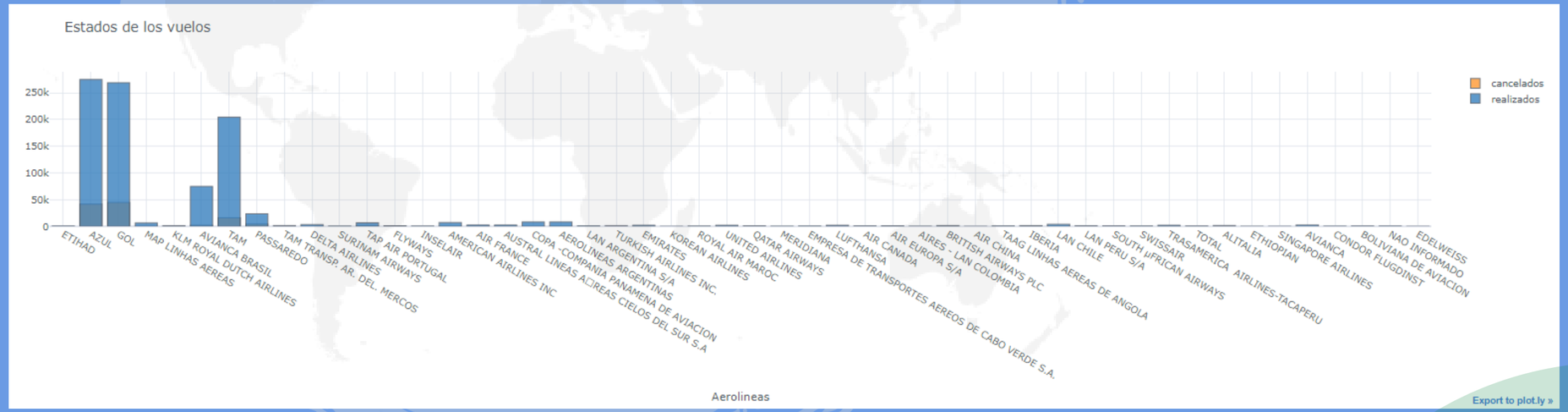
```
(1048575, 15)
```

Companhia.Aerea	Codigo.Tipo.Linha	Partida.Prevista	Partida.Real	Chegada.Prevista	Chegada.Real	Situacao.Voo	Codigo.Justificativa	Aeroporto.Origem	...	Pais.Origem	Aeroporto.Destino
ETIHAD	Internacional	2016-01-07 21:55:00	2016-01-07 21:59:00	2016-01-08 13:40:00	2016-01-08 13:01:00	Realizado	ATRASOS NAO ESPECIFICOS - OUTROS	Ab Dhabi International	...	Emirados Arabes Unidos	Guarulhos - Governador Andre Franco Montoro
ETIHAD	Internacional	2016-01-01 21:55:00	2016-01-02 08:36:00	2016-01-02 13:40:00	2016-01-02 14:21:00	Realizado	DEFEITOS DA AERONAVE	Ab Dhabi International	...	Emirados Arabes Unidos	Guarulhos - Governador Andre Franco Montoro
ETIHAD	Internacional	2016-01-28 21:55:00	2016-01-28 22:27:00	2016-01-29 13:40:00	2016-01-29 13:58:00	Realizado	ATRASOS NAO ESPECIFICOS - OUTROS	Ab Dhabi International	...	Emirados Arabes Unidos	Guarulhos - Governador Andre Franco Montoro

Dataset

Analizando algunas graficas decidimos no tomar en cuenta ciertos datos y modificar el dataset para tenerlo más limpio y poder hacer un estudio preciso y llegar a una conclusión favorable.

Graficas Analizadas



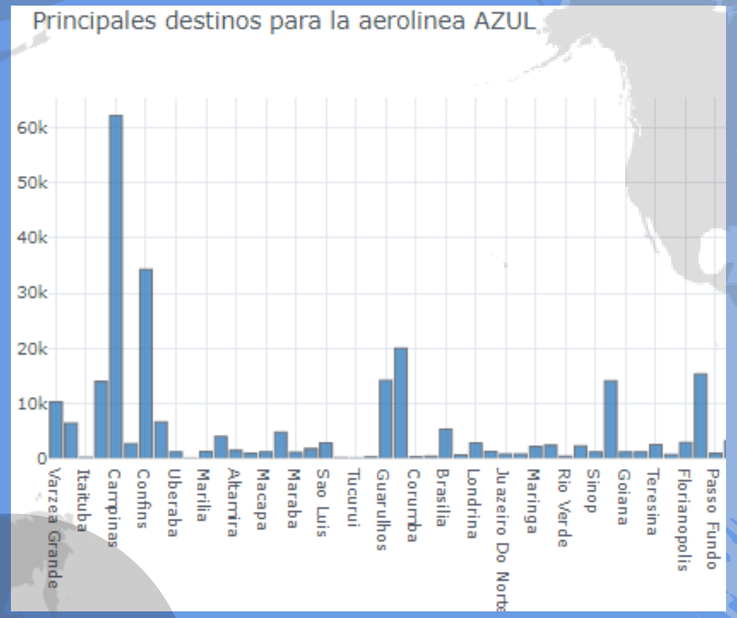
Podemos ver que hay 3 compañías aéreas en específico que tienen un gran porcentaje de vuelos realizados a comparación del resto, así que entraremos a ver las ciudades destino de estas 3 aerolíneas

Dataset

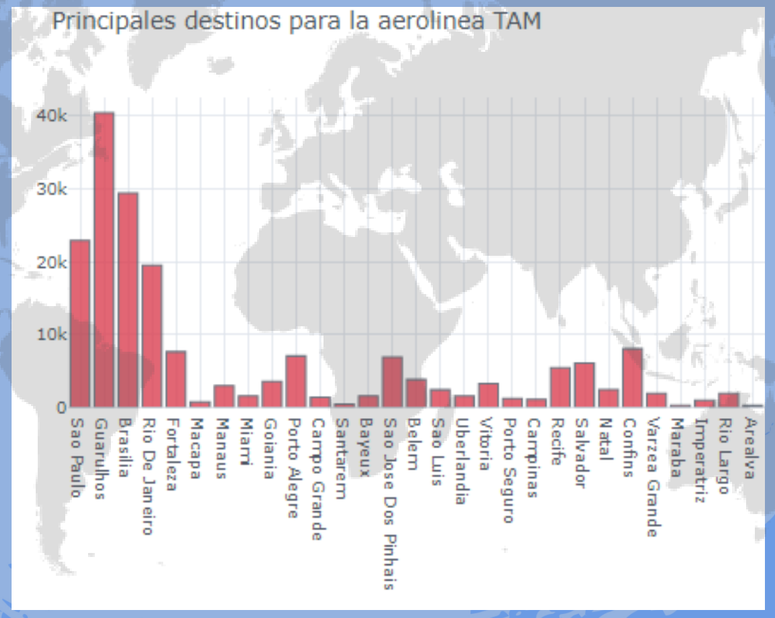
Graficamos los principales destinos de las 3 aerolíneas con mas vuelos realizados.

Graficas Analizadas

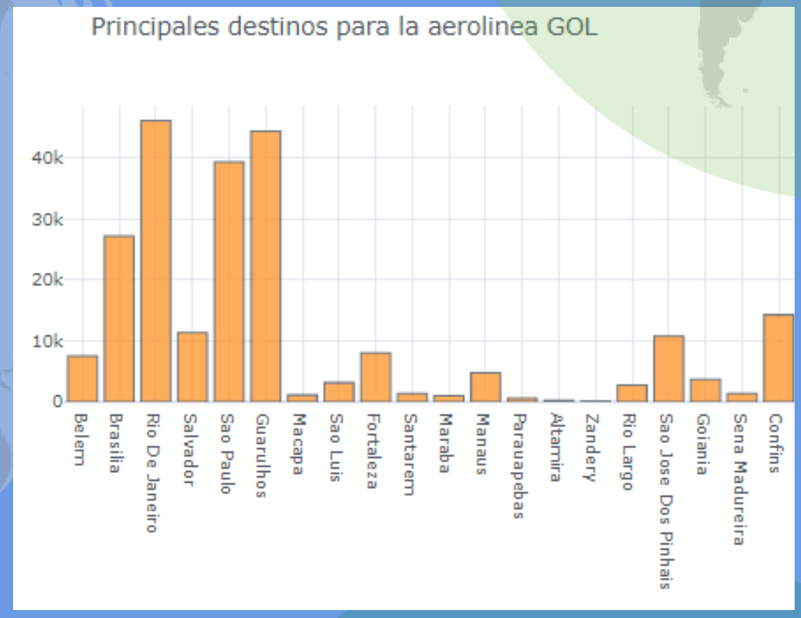
Aerolínea AZUL



Aerolínea TAM



Aerolínea GOL



Después de ver esta información se decidió volver todos estos datos a valores numéricos para poder trabajar con ellos otorgándole a cada ciudad y compañía un numero especifico, además reducimos el dataset para solo trabajar un par de ciudades y las compañías que vuela desde esos destinos.

Dataset

Después de reducir los datos y convertirlos a valores numéricos seleccionamos solo las columnas con las que vamos a trabajar y obtenemos como resultado este dataset:

	Voos	Companhia.Aerea	Codigo.Tipo.Linha	Cidade.Origem	Situacao.Voo	Codigo.Justificativa	tiempo de retraso
1030474	TAM - 8148	2.0	2	5.0	1	0.0	0
1031285	TAP - 102	7.0	2	2.0	1	0.0	0
1045954	UAE - 262	16.0	2	5.0	1	0.0	0
1047925	UAL - 860	17.0	2	5.0	0	2.0	-42369.795139

Donde la X serán las demás tablas y nuestra y será la tabla "Situacao.voo" con los siguientes valores:

Vuelo realizado : 1
Vuelo cancelado : 0

Las ciudades con las que vamos a trabajar ya que tienen un número considerable de viajes y están muy balanceados tendrán la siguiente clasificación:

Campinas : 1
Confins : 2
Rio de Janeiro : 3
Sao Paulo : 4
Guarulhos : 5
Brasilia : 6
Porto Alegre: 7

Estimadores para Clasificación

Teniendo ya el dataset óptimo para trabajar pasaremos a analizarlo con los estimadores.

Gaussian NB

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, r
est = GaussianNB()

est.fit(X_train,y_train)
y_pred = est.predict(X_test)
print("%.3f"%accuracy_score(est.predict(X_test), y_test))
print("%.3f"%accuracy_score(est.predict(X_train), y_train))
print(classification_report(y_test, y_pred))
```

```
0.994
0.994
              precision    recall  f1-score   support

           0       1.00      0.94      0.97     40179
           1       0.99      1.00      1.00    350160

    accuracy                           0.99    390339
   macro avg       1.00      0.97      0.98    390339
weighted avg       0.99      0.99      0.99    390339
```

Decision Tree

```
[22] from sklearn.tree import DecisionTreeClassifier
     from sklearn.model_selection import KFold
     from sklearn.model_selection import cross_val_score

     X_temp = a.values[:,1:4]
     y_count = a.values[:,4]
     y_count=y.astype('int')

     est = DecisionTreeClassifier(max_depth=2)
     est.fit(X_train,y_train)
     print(accuracy_score(est.predict(X_test), y_test))
```

```
0.993866900991697
```

Cross validation score del estimador

Cross validation score

```
s = cross_val_score(est, X, y, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score)
print (s)
print ("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
```

```
[0.99393863 0.99415383 0.99438696 0.99361584 0.99438696 0.9941359
 0.99354399 0.99415372 0.99352606 0.99402819]
accuracy 0.994 (+/- 0.00031)
```

Ramdom Forest

```
from sklearn.ensemble import RandomForestClassifier
est = RandomForestClassifier()
est.fit(X_train,y_train)
print(accuracy_score(est.predict(X_test), y_test))
```

```
0.9940013987769668
```

Clasificación por Deep Learning

Se decidió realizar una clasificación con Deep learning para predecir cuando hay probabilidad de que un vuelo se retrase o que salga a tiempo dependiendo de circunstancias alternas.

Red Neuronal

```
[ ] from sklearn.model_selection import train_test_split
    import tensorflow as tf
    from tensorflow import keras

    X = a.values[:,1:4]
    y = a.values[:,4]
    y=  y.astype(int)
    X=  X.astype(int)
    print( 'X:', X.shape)
    print( 'y:', y.shape)

    test_size = 0.4
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
    print(X_train.shape, X_test.shape)
    print(X_train[0].shape)
    print(y_train.shape, y_test.shape)

    y_train_ohe = tf.keras.utils.to_categorical(y_train, num_classes=4)
    y_test_ohe = tf.keras.utils.to_categorical(y_test, num_classes=4)
    print(y_train_ohe.shape, y_test_ohe.shape)

    model = tf.keras.Sequential([
        tf.keras.layers.Flatten( input_shape= X_train[0].shape),
        tf.keras.layers.Dense(128, activation='tanh'),
        tf.keras.layers.Dense(64,activation='tanh'),
        tf.keras.layers.Dense(32,activation='tanh'),
        tf.keras.layers.Dense(16,activation='relu'),
        tf.keras.layers.Dense(8,activation='relu'),
        tf.keras.layers.Dense(4, activation='sigmoid')     #softmax for multiple classes
    ])
    model.summary()

    model.compile(optimizer=tf.keras.optimizers.SGD(),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    model.fit(X_train, y_train_ohe, epochs=5)

    test_loss, test_acc = model.evaluate(X_test, y_test_ohe)

    print( 'Test accuracy:', test_acc, " test_loss: ", test_loss)
```

Se creo una red neuronal con 6 capas internas usando funciones como tanh, relu y sigmoid para la activación teniendo lo siguiente como resultado:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_11 (Flatten)        (None, 3)                 0

 dense_67 (Dense)            (None, 128)               512

 dense_68 (Dense)            (None, 64)                8256

 dense_69 (Dense)            (None, 32)                2080

 dense_70 (Dense)            (None, 16)                528

 dense_71 (Dense)            (None, 8)                 136

 dense_72 (Dense)            (None, 4)                 36

 dense_73 (Dense)            (None, 2)                 10

=================================================================
Total params: 11,558
Trainable params: 11,558
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
10456/10456 [==============================] - 20s 2ms/step - loss: 0.3294 - accuracy: 0.8960
Epoch 2/5
10456/10456 [==============================] - 19s 2ms/step - loss: 0.3211 - accuracy: 0.8975
Epoch 3/5
10456/10456 [==============================] - 19s 2ms/step - loss: 0.3181 - accuracy: 0.8975
Epoch 4/5
10456/10456 [==============================] - 19s 2ms/step - loss: 0.3153 - accuracy: 0.8975
Epoch 5/5
10456/10456 [==============================] - 19s 2ms/step - loss: 0.3135 - accuracy: 0.8975
6971/6971 [==============================] - 9s 1ms/step - loss: 0.3134 - accuracy: 0.8969
Test accuracy: 0.8969114422798157  test_loss:  0.31336379051208496
```

Clasificación por Deep Learning

Aquí tenemos una validación del accuracy para nuestra red neuronal