

Airline Passenger Satisfaction

(Hokages)





Table of contents



Introduction

Introduction, problem statement, objective



Dataset

Description, preparation, preprocessing, Exploration



PySpark ML

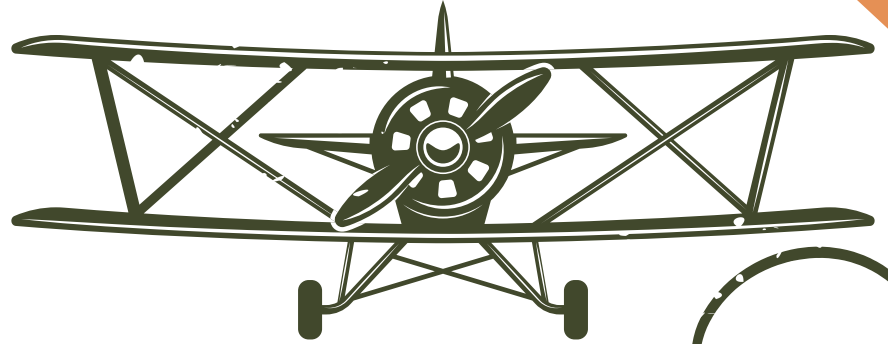
Processing, Building a Model, Evaluation





Introduction

Introduction, problem statement, objective



Introduction

Customer satisfaction plays a major role in affecting the business of a company therefore analyzing and improving the factors that are closely related to customer satisfaction is important for the growth and reputation of a company.



Problem Statement



One of the problems of many airline industries is how to measure customer satisfaction concerning the experience of using online services.

The objective or goal of this project is to guide an airline company to determine the important factors that influence customer or passenger satisfaction.





Dataset



Description, preparation, preprocessing, Exploration





Dataset Description

This dataset contains an airline passenger satisfaction survey:

- ▲ Which factors are highly correlated to a satisfied (or dissatisfied) passenger?
 - ▲ Can you predict passenger satisfaction?
- 
- 

About Dataset

Columns Description:

- Gender: Gender of the passengers (Female, Male)
- Customer Type: The customer type (Loyal customer, disloyal customer)
- Age: The actual age of the passengers
- Type of Travel: Purpose of the flight of the passengers (Personal Travel, Business Travel)
- Class: Travel class in the plane of the passengers (Business, Eco, Eco Plus)
- Flight distance: The flight distance of this journey



About Dataset

Columns Description:

- Inflight wifi service: Satisfaction level of the inflight wifi service (0:Not Applicable;1-5)
- Departure/Arrival time convenient: Satisfaction level of Departure/Arrival time convenient
- Ease of Online booking: Satisfaction level of online booking
- Gate location: Satisfaction level of Gate location
- Food and drink: Satisfaction level of Food and drink
- Online boarding: Satisfaction level of online boarding



About Dataset

Columns Description:

- Seat comfort: Satisfaction level of Seat comfort
- Inflight entertainment: Satisfaction level of inflight entertainment
- On-board service: Satisfaction level of On-board service
- Leg room service: Satisfaction level of Leg room service
- Baggage handling: Satisfaction level of baggage handling
- Check-in service: Satisfaction level of Check-in service



About Dataset

Columns Description:

- Inflight service: Satisfaction level of inflight service
- Cleanliness: Satisfaction level of Cleanliness
- Departure Delay in Minutes: Minutes delayed when departure
- Arrival Delay in Minutes: Minutes delayed when Arrival
- Satisfaction: Airline satisfaction level(Satisfaction, neutral or dissatisfaction)



Data Preparation

▼ Connect to the Spark server

```
✓ [3] spark = pyspark.sql.SparkSession.builder.getOrCreate()  
8s
```

▼ Obtain the Data

```
✓ [4] fullpath = 'Airline_Passenger_Satisfaction.csv'  
15s
```

```
data = spark.read.csv(fullpath)
```

```
data
```

```
DataFrame[_c0: string, _c1: string, _c2: string, _c3: string, _c4: string, _c5: string, _c6: string, _c7: string, _c8: string, _c9: string,  
_c10: string, _c11: string, _c12: string, _c13: string, _c14: string, _c15: string, _c16: string, _c17: string, _c18: string, _c19: string,  
_c20: string, _c21: string, _c22: string, _c23: string, _c24: string]
```

Data Preparation

✓
4s

```
[5] data = spark.read.csv(fullpath,  
                           sep=',',  
                           inferSchema=True,  
                           header=True,  
                           multiLine=True)  
  
data.printSchema()
```

```
data.printSchema()
```

```
root
|-- Unnamed: 0: integer (nullable = true)
|-- id: integer (nullable = true)
|-- Gender: string (nullable = true)
|-- CustomerType: string (nullable = true)
|-- Age: integer (nullable = true)
|-- TypeofTravel: string (nullable = true)
|-- Class: string (nullable = true)
|-- FlightDistance: integer (nullable = true)
|-- Inflighttwifiservice: integer (nullable = true)
|-- Departure/Arrivaltimeconvenient: integer (nullable = true)
|-- EaseofOnlinebooking: integer (nullable = true)
|-- Gatelocation: integer (nullable = true)
|-- Foodanddrink: integer (nullable = true)
|-- Onlineboarding: integer (nullable = true)
|-- Seatcomfort: integer (nullable = true)
|-- Inflightentertainment: integer (nullable = true)
|-- On-boardservice: integer (nullable = true)
|-- Legroomservice: integer (nullable = true)
|-- Baggagehandling: integer (nullable = true)
|-- Checkinservice: integer (nullable = true)
|-- Inflightservice: integer (nullable = true)
|-- Cleanliness: integer (nullable = true)
|-- DepartureDelayinMinutes: integer (nullable = true)
|-- ArrivalDelayinMinutes: double (nullable = true)
|-- satisfaction: integer (nullable = true)
```

Data Preprocessing



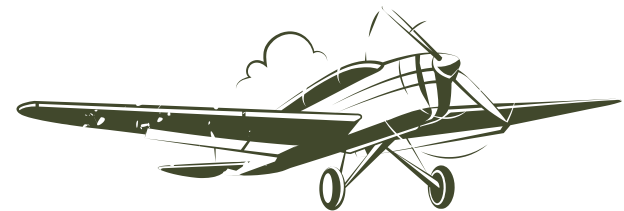
Data Cleaning

```
# these columns are useless to us, drop them
drop_cols = ['DepartureDelayinMinutes', 'ArrivalDelayinMinutes', '_c0', 'id']

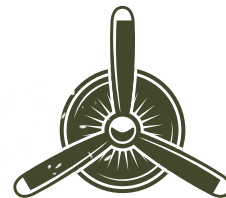
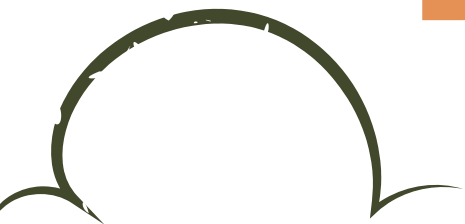
data = data.drop(*drop_cols)

data = data.replace('other', None, subset=['Gender'])

data = data.replace('other', None, subset=['Class'])
```

Data Exploration



```
data.registerTempTable('data')

state_counts = spark.sql(r"""SELECT Gender, COUNT(Gender) AS total
                             FROM data
                             GROUP BY Gender
                             ORDER BY total desc """)

state_counts.show()
```

```
+-----+-----+
|Gender|total|
+-----+-----+
|Female|52727|
|  Male|51177|
+-----+-----+
```

```
mfr_counts = spark.sql(r"""SELECT Class, COUNT(Class) as total
                             FROM data
                             GROUP BY Class
                             ORDER BY total desc""")
```

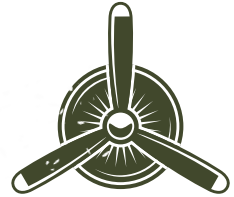
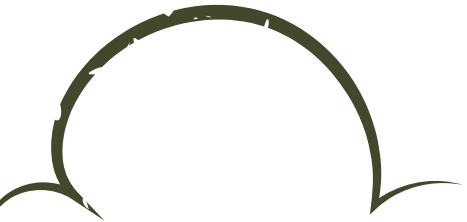
```
mfr_counts.show()
```

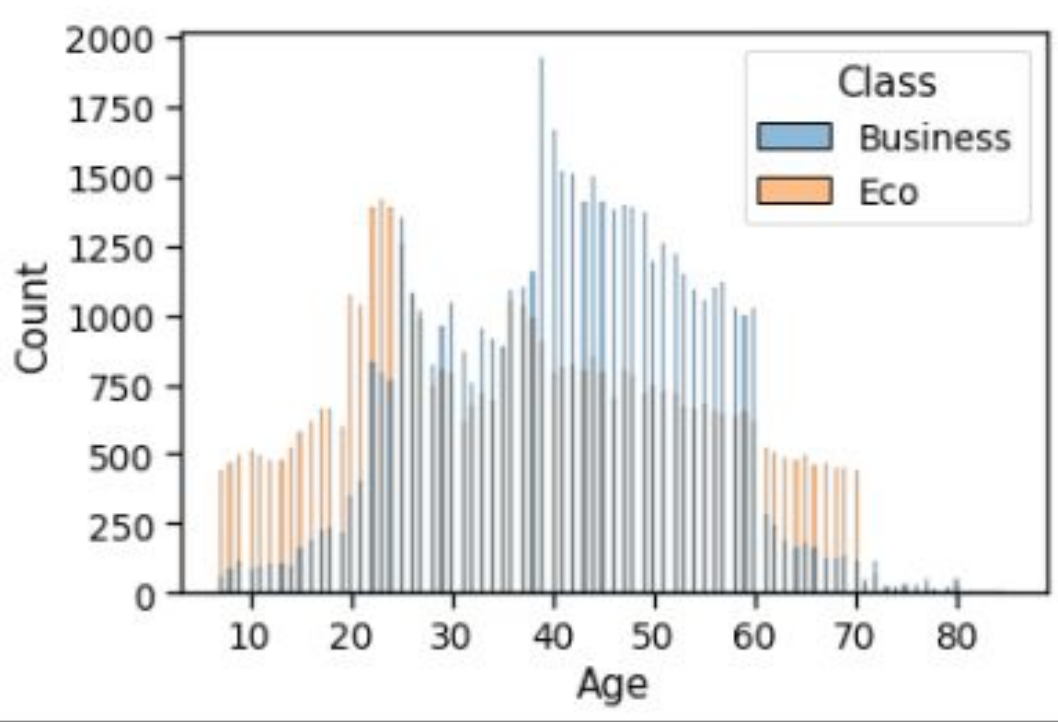
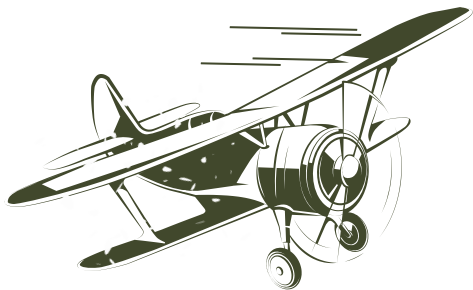
```
+-----+-----+
|  Class|total|
+-----+-----+
|Business|49665|
|    Eco|46745|
|Eco Plus| 7494|
+-----+-----+
```

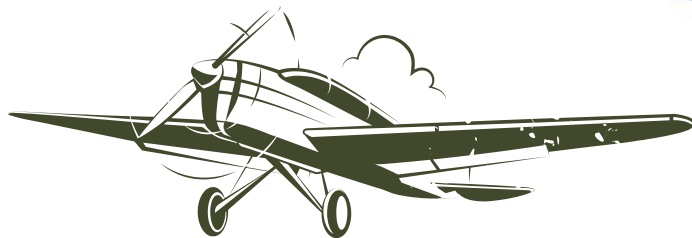
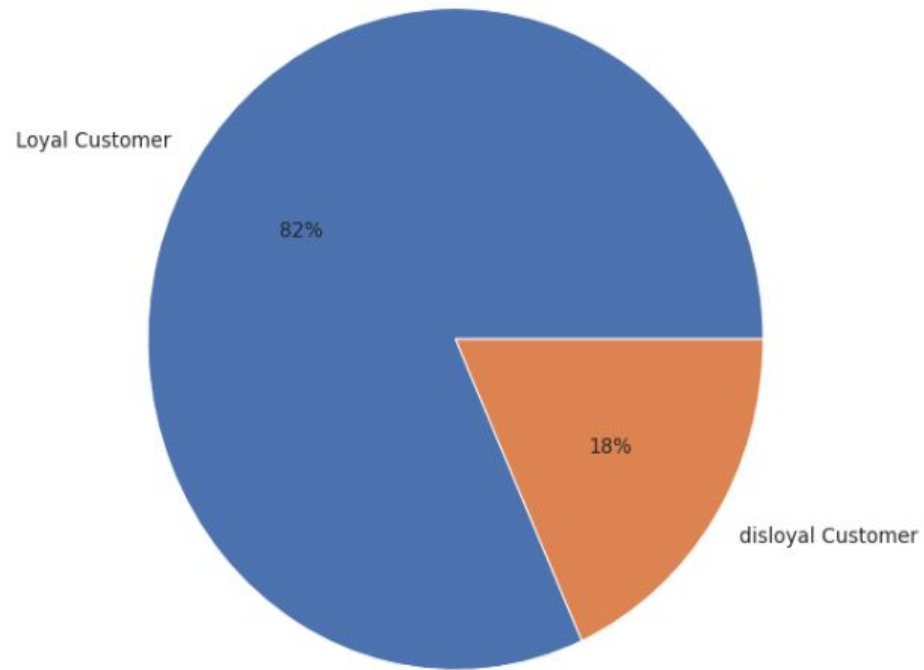
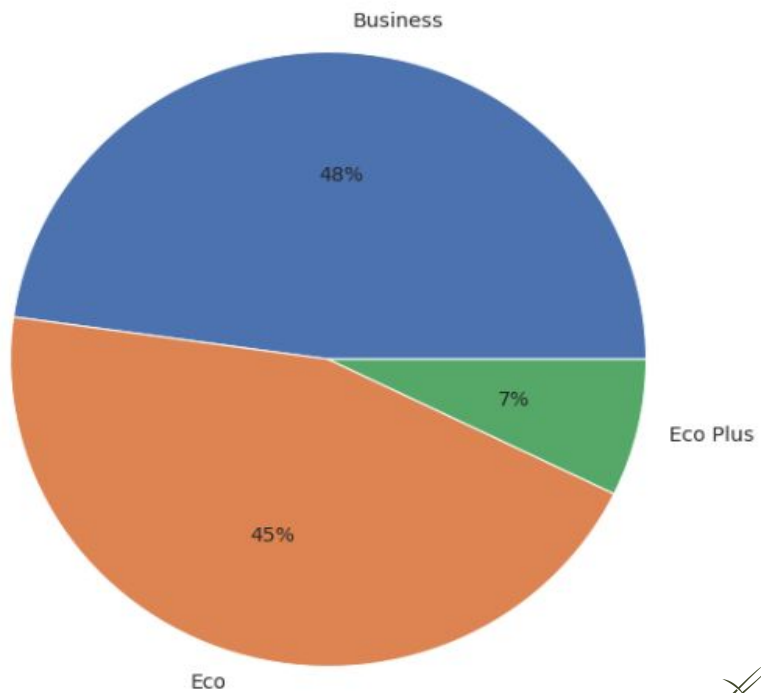




Exploratory Data Analysis (EDA)



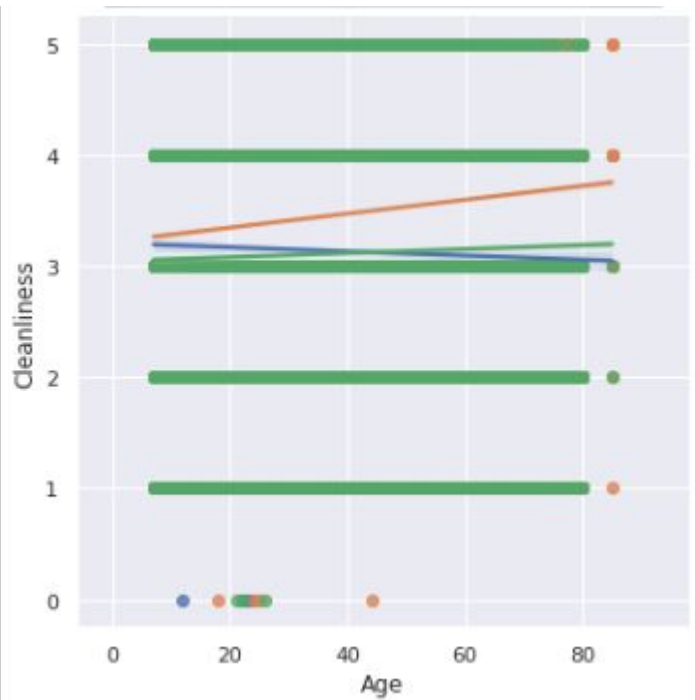






Class

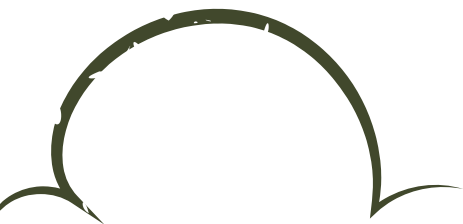
- Eco Plus
- Business
- Eco



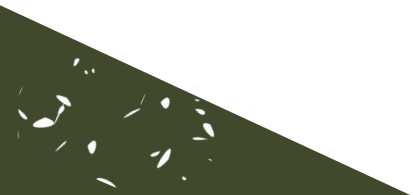


PySpark ML

Processing, Building a Model, Evaluation



Data Preparation



Category columns

```
cat_cols = ['Gender', 'CustomerType', 'TypeofTravel', 'Class', 'satisfaction']  
  
n = 4  
  
for col in cat_cols:  
    most_freq = data.groupBy(col).count().orderBy('count', ascending=False).take(n - 1)  
    most_freq = spark.createDataFrame(most_freq).toPandas()  
    most_freq = most_freq[col].tolist()  
  
    data = data.withColumn(col, F.when(F.col(col).isin(most_freq), F.col(col)))
```





On Categorical Columns, we will encode all the categorical columns using StringIndexer and drop the original columns.

```
[ ] for col in cat_cols:
    indexer = StringIndexer(inputCol=col, outputCol=col+'_idx')
    data = indexer.fit(data).transform(data)

data = data.drop(*cat_cols)
```





```
cols = data.columns  
cols.remove('satisfaction_idx') #remove -> we need this to be our label
```

```
assembler = VectorAssembler(inputCols=cols, outputCol='features')
```

```
data = assembler.transform(data)
```

```
# We have created a new dataframe only consisting of the features column and the label column (actually price column but renamed)  
df_data = data.select(F.col('features'), F.col('satisfaction_idx').alias('label'))
```

```
df_train, df_test = df_data.randomSplit([0.8, 0.2])
```



Building a Model





Building a Model

▼ Model Building

```
▶ evaluator = RegressionEvaluator() # Can specify what metrics we want to use. Default metric is Root Mean Squared Error (RMSE)
grid = ParamGridBuilder().build()
```

▼ Initialize Regressors and Train

```
[ ] #Random Forest Regressor
classifier_rf = RandomForestRegressor(featuresCol='features', labelCol='label')
cv_rf = CrossValidator(estimator=classifier_rf, evaluator=evaluator, estimatorParamMaps=grid, numFolds=5)
cv_model_rf = cv_rf.fit(df_train)
```

```
[ ] #Gradient Boosted Tree Regressor
classifier_gbt = GBTRegressor(featuresCol="features", labelCol='label', maxIter=10)
cv_gbt = CrossValidator(estimator=classifier_gbt, evaluator=evaluator, estimatorParamMaps=grid, numFolds=5)
cv_model_gbt = cv_gbt.fit(df_train)
```

```
[ ] #Linear Regression
classifier_lr = LinearRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)
cv_lr = CrossValidator(estimator=classifier_lr, evaluator=evaluator, estimatorParamMaps=grid, numFolds=5)
cv_model_lr = cv_lr.fit(df_train)
```

Model Evaluation





Model Evaluation

```
metrics = []
models = [cv_model_rf, cv_model_gbt, cv_model_lr]

for model in models:
    metrics.append(model.avgMetrics)
print (metrics)

for idx, model in enumerate(models):
    metrics[idx].append(RegressionEvaluator(predictionCol='prediction', labelCol='label', metricName='r2').evaluate(model.bestModel))
    metrics[idx].append(RegressionEvaluator(predictionCol='prediction', labelCol='label', metricName='rmse').evaluate(model.bestModel))
    metrics[idx].append(RegressionEvaluator(predictionCol='prediction', labelCol='label', metricName='mae').evaluate(model.bestModel))

df = pd.DataFrame(metrics, index = ['Random Forest Regressor', 'Gradient Boosted Tree Regressor', 'Linear Regression'], columns=
df
[[0.24528405092644592], [0.22446513982488212], [0.4954913818117944]]
```

	Average Metrics (CV)	Best Model R2 on Test Set	Best Model RMSE on Test Set	Best Model MAE on Test Set
Random Forest Regressor	0.245284	0.752917	0.246414	0.162691
Gradient Boosted Tree Regressor	0.224465	0.795455	0.224201	0.113253
Linear Regression	0.495491	-0.000013	0.495732	0.491257

Thanks!

Do you have any questions?

Done by:

- ▲ Shouq Alharbi
- ▲ Razan Alajlan
- ▲ Nada Oteif
- ▲ Hayam Alrashed
- ▲ Sarah Alrashidi

