Data Explorin' the World

Rachel Roggenkemper and Lana Huynh

Research Questions and Motivation

Motivation:

- Both of us are enthusiastic travelers and are thrilled to be looking into more travel related data

Research Questions:

1. What does tourism look like on an international level?
2. What does flying etiquette look like to different groups of people and can we predict whether or not a passenger reclines their seat on flights?

01

Data Collection and Cleaning

Flying Etiquette Data

- SurveyMonkey Audience poll done by FiveThirtyEight
- 1,040 respondents, 874 of whom had flown

```
RespondentID
How often do you travel by plane?
Do you ever recline your seat when you fly?
How tall are you?
Do you have any children under 18?
In a row of three seats, who should get to use the two arm rests?
In a row of two seats, who should get to use the middle arm rest?
Who should have control over the window shade?
Is itrude to move to an unsold seat on a plane?
```

International Tourism Data

Observational unit: International tourism data for each country for each year (years ranging from 1960 to 2021) -> filtered data to focus on 1995 to 2017 to reduce NA values and to filter out projected data from Gapminder

The World Bank:

- International tourism, expenditures (% of total imports)
- International tourism, expenditures (current US$, inflation-adjusted)
- International tourism, expenditures for passenger transport items (current US$, inflation-adjusted)
- International tourism, number of arrivals
- International tourism, number of departures
- International tourism, receipts (current US$, inflation-adjusted)
- International tourism, receipts for passenger transport items (current US$, inflation-adjusted)
- International tourism, receipts for travel items (current US$, inflation-adjusted)
- International tourism, receipts (% of total exports)

Gapminder:

- CO2 emissions (tonnes per person)
- Income per person (GDP/capita, PPP$ inflation-adjusted)
- Population
- Total GDP (US$, inflation-adjusted)

Country -> Continent:

- Subregion and continent of each country

02

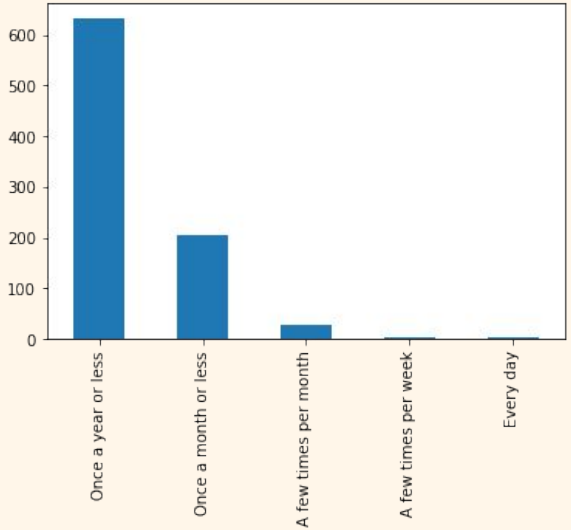
Data Exploration

Flying Etiquette Data Overview

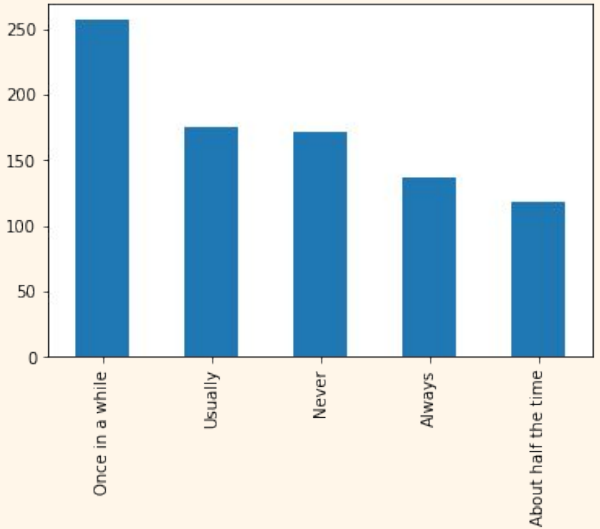
ASD

05-36

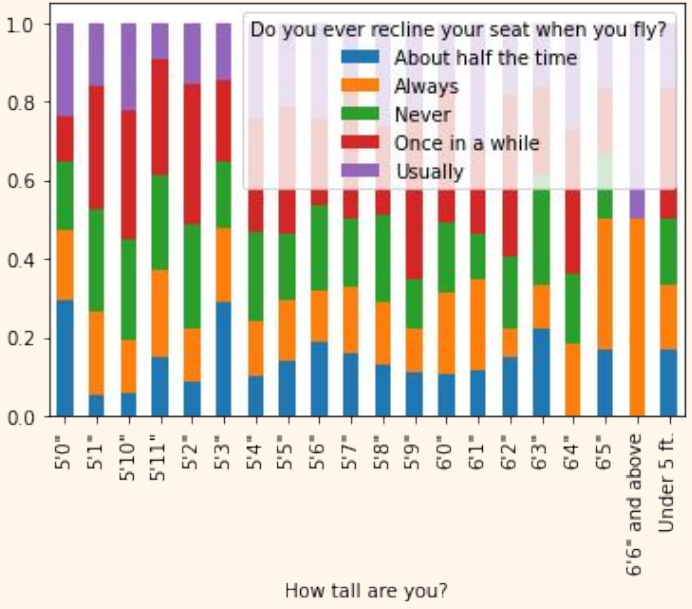
How often do you travel by plane?



Do you ever recline your seat when you fly?

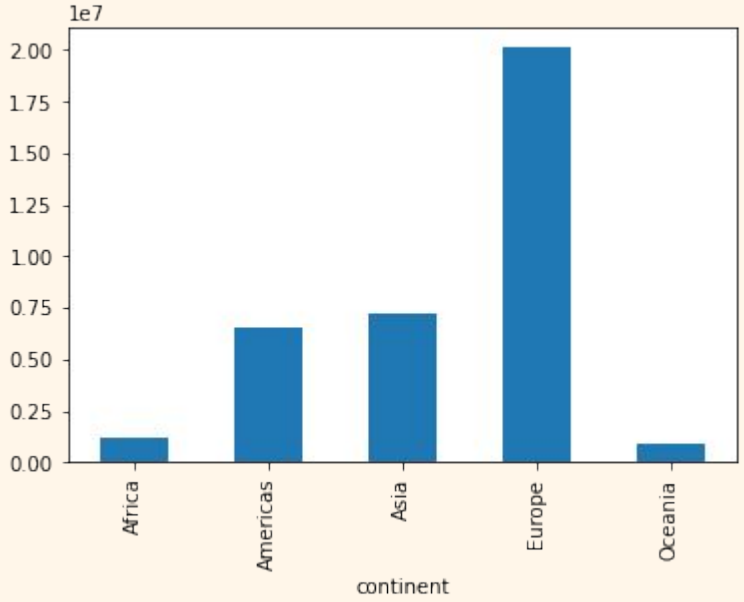


Relationship Between Height and Reclining Seat

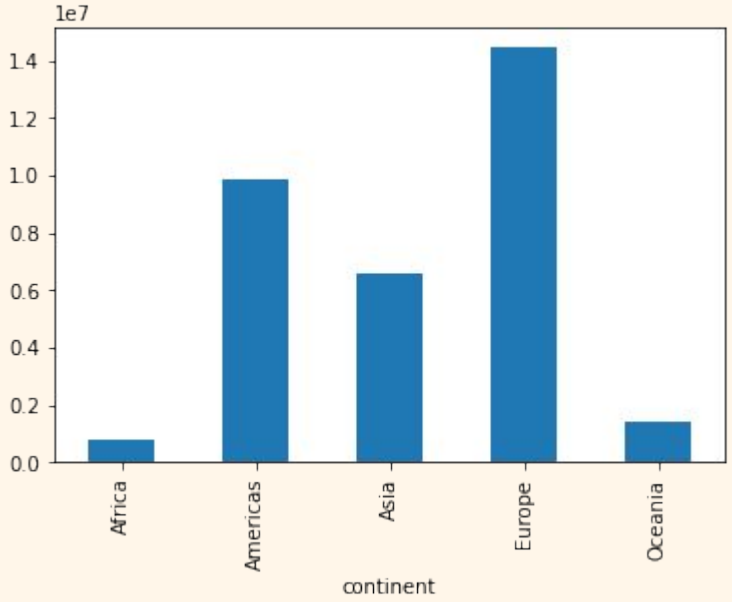


Arrivals and Departures

Number of Arrivals by Continent



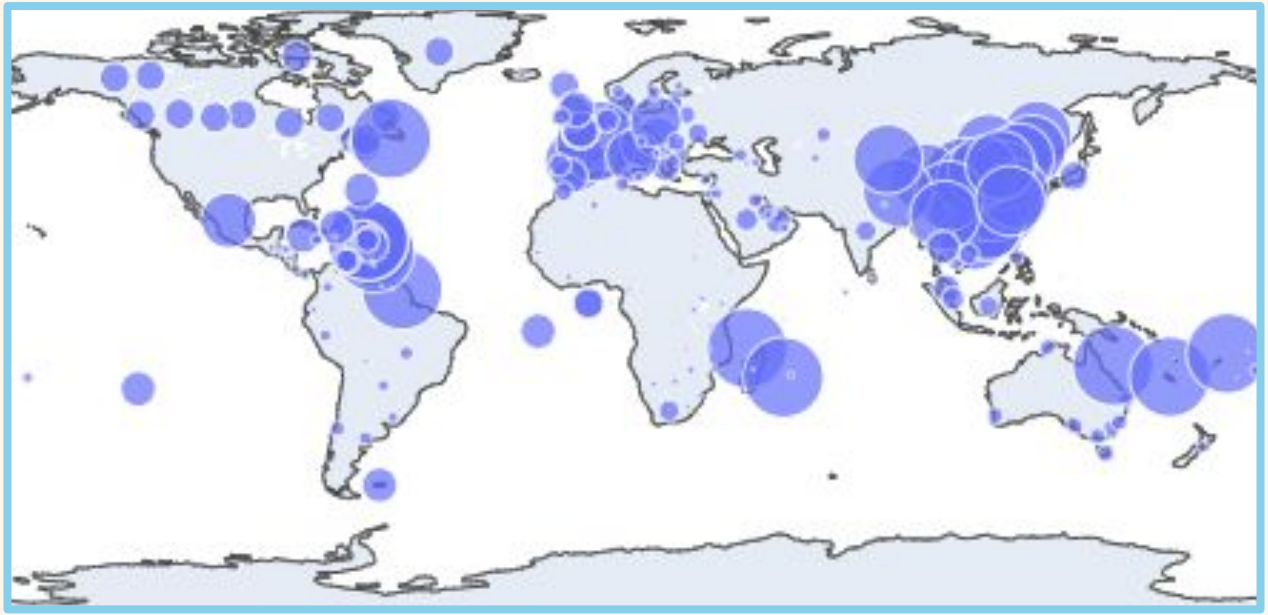
Number of Departures by Continent



ASD

05-36

Number of Arrivals: Dot Map



03

Machine Learning

KNN Model: Flying Etiquette Data

- Built a k-nearest neighbors model to predict if passengers ever recline their seat when they fly
- Predictor variables:
  - Height (Quantitative)
  - Age (Categorical)
- Ended up with an accuracy of 23%, yikes!

```
Once in a while
Usually
Never
Always
About half the time
```

```
ct_flyet = make_column_transformer(
    (OneHotEncoder(sparse=False), ["Height",
                                   "Age"]),
    remainder='passthrough'
)

knn_model = make_pipeline(
    ct_flyet,
    KNeighborsClassifier(n_neighbors=5)
)

knn_model.fit(X_train, y_train)
```

KNN Model: Flying Etiquette Data

- Wrote a function to turn our response variable to a binary one
- Optimized the number of neighbors using a grid search and ended up with k = 21
- Accuracy is now 81%, yay!

```
def recline_binary(s) -> int:
  if s == "Never":
    return "No"
  return "Yes"
```

Yes	558
No	129

```
grid_search = GridSearchCV(
    knn_model,
    param_grid={"kneighborsclassifier__n_neighbors": range(1, 50)},
    scoring="accuracy",
    cv=10
)

grid_search.fit(X_train, y_train)
grid_search.best_params_
```

```
{'kneighborsclassifier__n_neighbors': 21}
```

Testing out the model on ourselves

- Defined a new dataframe to use the machine learning algorithm on ourselves
  - Lana - index 0
  - Rachel - index 1
- Both predictions were accurate because we recline our seats on flights!! :)

```
# define the test data (recall that scikit-learn expects a 2D-array)
x_new = pd.DataFrame(columns=["Height",
                              "Age"])
x_new.loc[0] = [63, "18-29"]
x_new.loc[1] = [69, "18-29"]
knn_model.predict(x_new)
```

```
array(['Yes', 'Yes'], dtype=object)
```

Ensemble Model: International Tourism

- Tried to predict CO2 emission levels
- Tested out a number of travel-related predictor variables

```
# calculate estimate of test error for a given feature set
def get_cv_error(features):
  # define pipeline
  pipeline = make_pipeline(
      StandardScaler(),
      KNeighborsRegressor(n_neighbors=5)
  )
  # calculate errors from cross-validation
  cv_errs = -cross_val_score(pipeline, X=df_2017[features],
                             y=df_2017["co2_emissions"],
                             scoring="neg_mean_squared_error", cv=3)
  # calculate average of the cross-validation errors
  return cv_errs.mean()

# calculate and store errors for different feature sets
errs = pd.Series()
for features in [["income_per_person", "population", "receipts_travel_items"],
                 ["income_per_person", "population", "number_of_departures"],
                 ["income_per_person", "population", "receipts_current_us"],
                 ["income_per_person", "population", "receipts_percent_total_exports"],
                 ["income_per_person", "population", "expenditures_percent_total_imports"]]:
  errs[str(features)] = get_cv_error(features)

errs
```

Ensemble Model: International Tourism

- Ensemble model produced the lowest cross-validation score (still pretty high)
- RMSE is fairly high, but this tells us valuable information that our model is not the best despite seemingly “linear trends” in the data exploration steps

```
for model in [linear_model, knn_model, ensemble_model]:
    print(-cross_val_score(model, X=X_train, y=y_train, cv=5,
                           scoring="neg_mean_squared_error").mean())
```

```
12.295825712894542
14.982486225396823
11.767698616704475
```

```
# get model's predictions on validation set
y_predict = ensemble_model.predict(X_train)

# calculate RMSE on validation set
rmse = np.sqrt(mean_squared_error(y_train, y_predict))
rmse
```

```
2.721978072758195
```

Thank you
Gracias
Merci
Danke
Xie xie
Cảm ơn
Arigato