

# IBM Applied Data Science Capstone Project

Capstone Project by  
*Mohammad Ashraf Ansari*

# Stakeholder Presentation

- Problem Statement
- Dataset
- Use cases
- The Solution

# Predict average spend on Food & Beverages by a holiday resort member

## Problem Statement

**The company, the most preferred holiday resort chain in India** makes significant revenue from Food and Beverages (F&B) sales in their resorts. The members of the company club are offered a wide variety of items in either buffet or restaurant form.

**Given the information related to resort, club member, reservation etc. the task is to predict average spend per room night on food and beverages for each reservation.**

**This is a regression problem, and we will train a machine learning model on the provided data to give the predictions.**

The problem statement and data were provided by Club Mahindra at online hackathon DataOlympics in association with Analytics Vidhya:

<https://datahack.analyticsvidhya.com/contest/club-mahindra-dataolympics/>

# The Dataset

## Dataset info

<b>Number of variables</b>	24
<b>Number of observations</b>	314 424
<b>Total Missing (%)</b>	0.1%

# Variable types

<b>Numeric</b>	11
<b>Categorical</b>	14
<b>Date</b>	3

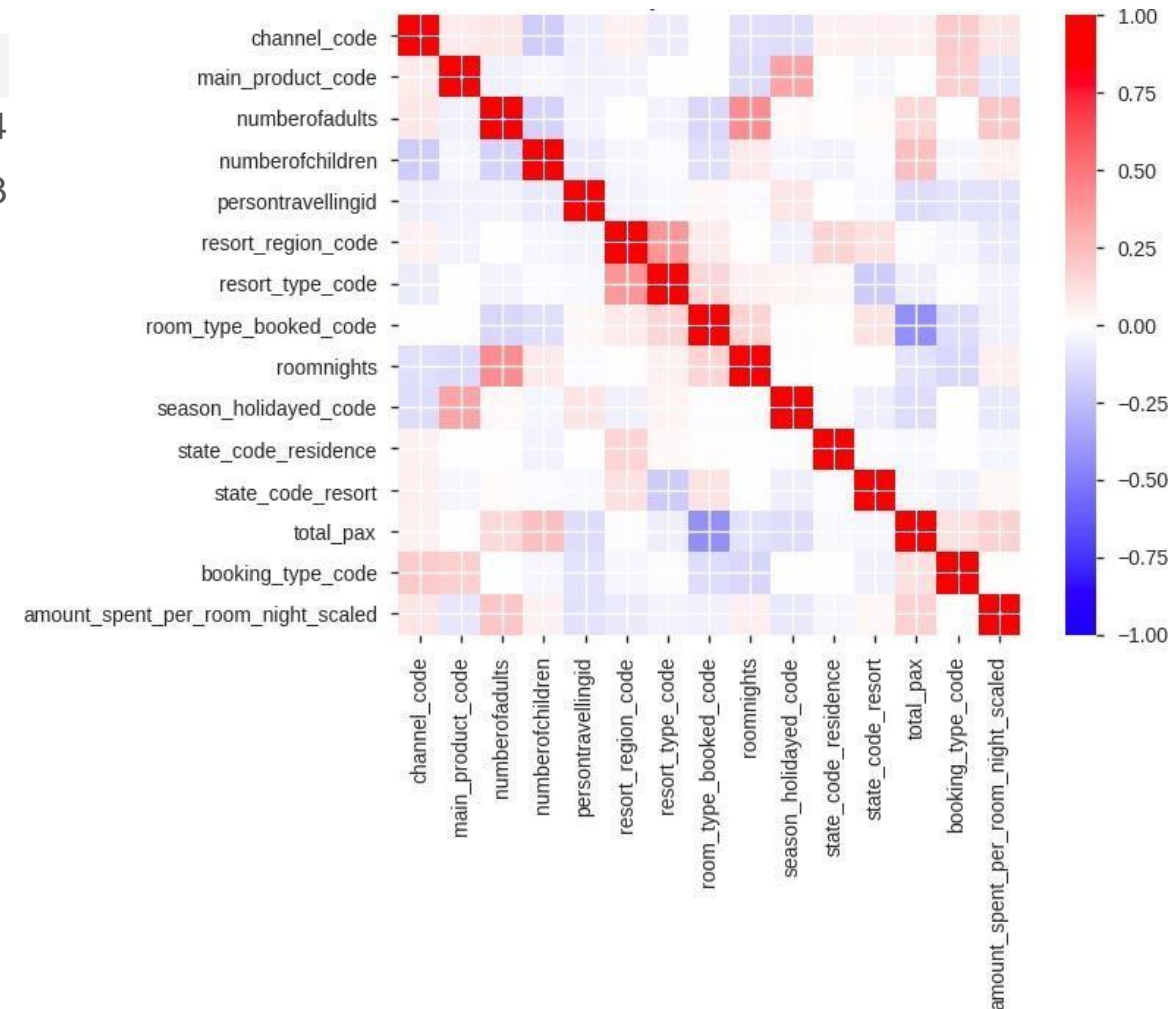
**The variables have been anonymized:**

- Categories (resort, state etc.) replaced with codes.
- Target variable (amount spent per room night) scaled.

### The variables most correlated with the target:

- `total_pax` – total number of persons travelling
- `numberofadults` – number of adults travelling
- `roomnights` – number of room nights booked
- `channel_code` – different channels for booking
- `state code resort` – state in which resort is located

## Correlation Matrix



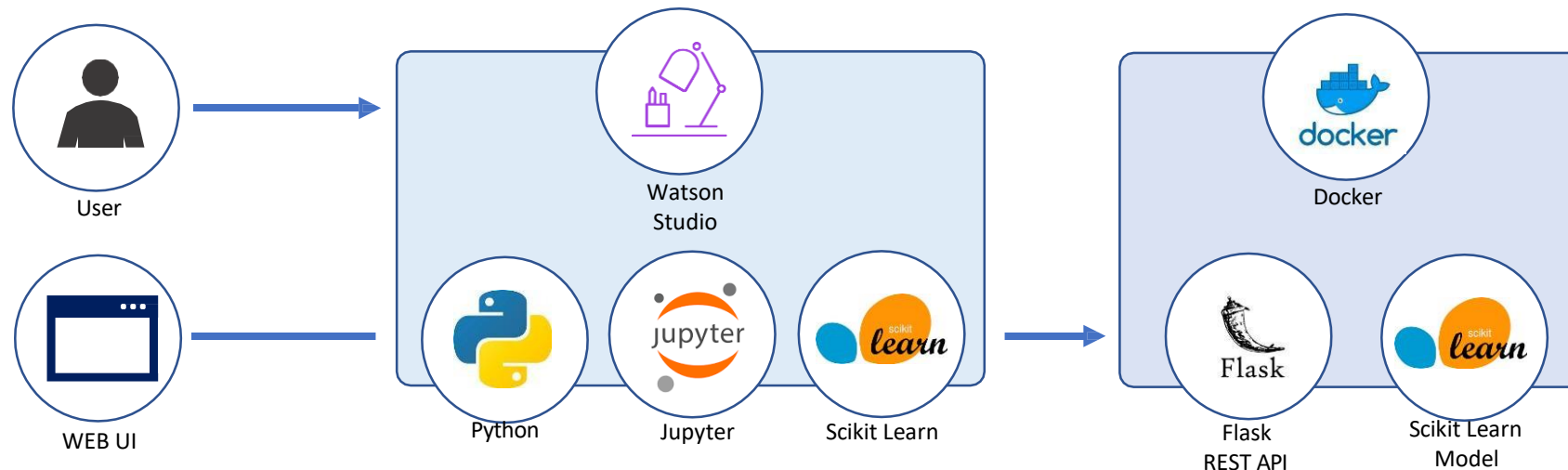
# The Use Case and the Solution

**Following are some benefits the prediction model will bring:**

- Predicting the F&B spend of a member in a resort would help in improving the pre-sales during resort booking through web and mobile app.
- Targeted campaigns to suit the member taste and preference of F&B.
- Providing members in the resort with a customized experience and offers.
- Help resort kitchen to plan the inventory and food quantity to be prepared in advance.

**The data preparation and model training is conducted with Watson Studio. It uses Python, scikit learn and Jupyter technology. And deployed as a web-service with IBM Cloud.**

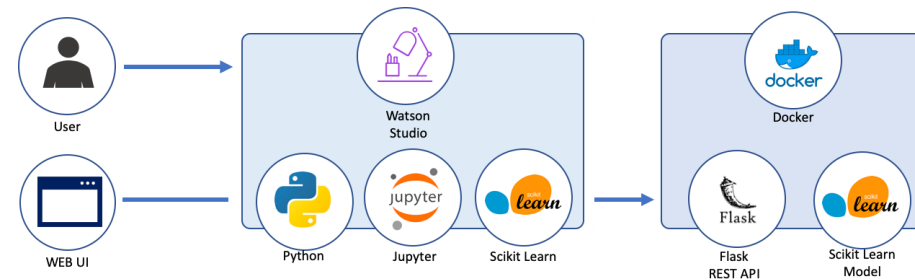
**The solution architecture:**



# Technical Presentation

- Architectural choices
- Data quality assessment, data pre-processing and feature engineering
- Model performance indicators
- Model algorithm

# Architectural Choices



## Core solution:

User provided CSV file uploaded to Watson Studio / IBM Object Storage.

Scikit-learn and Python in IBM Watson Studio with Jupyter notebooks:

- ETL; Data Exploration.
- Feature Engineering; Model Definition and Validation.
- Pipeline Notebook (takes raw CSV data and outputs the trained pipeline ready for deployment).

**Trained model is saved and deployed** as Flask REST API with Dash UI hosted at IBM Cloud.

## Alternative solution:

- IBM Watson Studio Data refinery flow for raw data preparation.
- Watson Machine Learning automatic modeling deployed as REST API web-service.

# Data Pre-Processing and Feature Engineering

## EDA

- 24 variables and 314 424 observations (unique reservations).
- 0.1% of total values are missing, 1.4% of missing values in `resort_type_code`.
- `booking_date`, `checkin_date` and `checkout_date` are date variables and should be parsed to datetime format at data loading stage.
- `memberid` has high cardinality: 101 327 unique values.
- `numberofchildren` has 77.4% of zero values.

## Feature Engineering

- **Date features:**  
`booking_in_advance`,  
`days_stayed`
- Other features containing the information from the date (`month`, `week`, `dayofweek`, `is_month_start` etc.)
- **Other features** (`n_people` etc.)

## Pre-processing

- Fill NA values with -1
- Convert binary-type columns values to [0, 1]
- OHE for categorical features
- Scale the numerical features



# Model Performance Indicators

The model is evaluated on Root-Mean-Squared-Error (RMSE) between the predicted value and the observed amount spent.

The train/valid split performed based on `memberid` variable to prevent the data leak: we want to be able to predict the amount spent for the new members of hotels.

```
# shuffle data
random.seed(42)
df = df.iloc[random.sample(range(len(df)), len(df)), :]

# get groups
groups = df["memberid"].values

# get indices for train and validation data
group_shuffle_split = GroupShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
train_idx, valid_idx = next(group_shuffle_split.split(df, groups=groups))

# split data
train_df = df.iloc[train_idx, :]
valid_df = df.iloc[valid_idx, :]
```

# Model Algorithm

The chosen model is **Linear Regression** with L2 regularization (Ridge Regression).

Model parameters are tuned on 5-fold cross validation stratified by `memberid` groups.  
The model performance is then validated with unseen data:

```
%%time
gkf = list(GroupKFold(n_splits=5).split(train_df, groups=groups))

model = RidgeCV(cv=gkf, scoring="neg_mean_squared_error")

model.fit(train, train_df['amount_spent_per_room_night_scaled'])

CPU times: user 50 s, sys: 21.4 s, total: 1min 11s
Wall time: 1min 8s

valid_score = np.sqrt(mean_squared_error(valid_df['amount_spent_per_room_night_scaled'], model.predict(valid)))

print("Best alpha: {} \nRidge RMSE Validation Score: {:.4f}".format(model.alpha_, valid_score))

Best alpha: 1.0
Ridge RMSE Validation Score: 0.9157
```

The model is trained and cross-validated in under 2 minutes with Watson Studio Lite plan (1 vCPU and 4 Gb RAM) and shows **RMSE score of 0.9157**.

# Model Deployment

After model training and evaluation, we create separate Jupyter notebook with full pipeline:

1. Loading data.
2. Pre-processing.
3. Create and save sklearn pipeline object to pickle file:
  - Choose columns
  - Transform columns
  - Train the model

```
pipeline = Pipeline([
    ('column_dropper', cs),
    ('mapper_transformer', mapper),
    ('ridge_model', Ridge(alpha=1, random_state=42))
])
```

```
%%time
pipeline.fit(df.drop(['amount_spent_per_room_night_scaled'], axis=1), df['amount_spent_per_room_night_scaled'])
```

```
CPU times: user 28.5 s, sys: 35.6 s, total: 1min 4s
Wall time: 1min 5s
```

```
Pipeline(memory=None,
      steps=[('column_dropper', ColumnSelector(columns=['channel_code', 'main_product_code', 'numberofadults', 'number
ofchildren', 'persontravellingid', 'resort_region_code', 'resort_type_code', 'room_type_booked_code', 'roomnights',
'season_holidayed_code', 'state_code_residence', 'state_code_resort', 't... fit_intercept=True, max_iter=None,
      normalize=False, random_state=42, solver='auto', tol=0.001))])
```

Save pipeline to pickle

```
with open('pipeline.pickle', 'wb') as pkl:
    pickle.dump(pipeline, pkl, protocol=pickle.HIGHEST_PROTOCOL)

cos.upload_file(Filename='pipeline.pickle', Bucket=credentials['BUCKET'], Key='pipeline.pickle')
```

The fitted and trained pipeline is then deployed as Flask REST API service with minimalist web-interface created with Dash, running at IBM Cloud.