Step no 9 :- Building a Marketing Mix Model

To get a better intuition for marketing mix models, this section will walk through building a marketing mix model from scratch in Python.

A: Import all relevant libraries and data.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as pltdf = pd.read_csv("../input/advertising.csv/Advertising.csv")
```

B : Perform some EDA

conduct a lot more exploratory data analyses, but for this tutorial we'll focus on the three most common (and powerful, in my experience):

1. **Correlation matrices**: a table that shows the correlation values for each pair-relationship
2. **Pair plots**: a simple way to visualize the relationships between each variable
3. **Feature importance**: techniques that assign a score for each feature based on how useful they are at predicting the target variable

Correlation Matrix

To reiterate, a correlation matrix is a table that shows the correlation values for each pair-relationship. It’s a very fast and efficient way of understanding feature relationships. Here’s the code for our matrix.

```
corr = df.corr()
sns.heatmap(corr, xticklabels = corr.columns, yticklabels = corr.columns, annot = True, cmap =
sns.diverging_palette(220, 20, as_cmap=True))
```

Pair plot

A pair plot is a simple way to visualize the relationships between each variable — it’s similar to a correlation matrix except it shows a graph for each pair-relationship instead of a correlation. Now let’s take a look at the code for our pair plot

```
sns.pairplot(df)
```

Feature Importance

Feature importance allows you to determine how “important” each input variable is to predict the output variable. A feature is important if shuffling its values increases model error because this means the model relied on the feature for the prediction.

```
# Setting X and y variables
X = df.loc[:, df.columns != 'sales']
y = df['sales']# Building Random Forest modelfrom sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error as maeX_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=.25, random_state=0)
model = RandomForestRegressor(random_state=1)
model.fit(X_train, y_train)
pred = model.predict(X_test)# Visualizing Feature Importance
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(25).plot(kind='barh',figsize=(10,10))
```

C : Build the Marketing Mix Model (aka. OLS model)

It’s time to build our marketing mix model! Another way to refer to the model we’re building is an OLS model, short for ordinary least squares, which is a method used to estimate the parameters in a linear regression model. An OLS model is a type of regression model that is most commonly used when building marketing mix models.

```
import statsmodels.formula.api as smmodel = sm.ols(formula="sales~TV+radio+newspaper",
data=df).fit()print(model.summary())
```

D : Plot Actual vs Predicted Values

Next, let’s graph the predicted sales values with the actual sales values to visually see how our model performs. This is a particularly useful thing to do in a business use case if you’re trying to see how well your model reflects what’s actually happening — in this case, if you’re trying to see how well your model predicts sales based on the amount spent in each marketing channel.

```
from matplotlib.pyplot import figurey_pred = model.predict()
labels = df['sales']
df_temp = pd.DataFrame({'Actual': labels, 'Predicted':y_pred})
df_temp.head()figure(num=None, figsize=(15, 6), dpi=80, facecolor='w', edgecolor='k')
y1 = df_temp['Actual']
y2 = df_temp['Predicted']plt.plot(y1, label = 'Actual')plt.plot(y2, label = 'Predicted')
plt.legend()
plt.show()
```

How to Interpret a Marketing Mix Model

Going back to the output from .summary(), there are a couple of things to focus on:

1. ***.summary()*** provides us with an abundance of insights on our model. Going back to the output from *.summary()*, we can see a few areas to focus in on (you can reference these insights against the OLS regression results below):

2. **The Adj. R-squared is 0.896**. This means that approximately 90% of the total variation in the data can be explained by the model. This also means that the model doesn't account for 10% of the data used — this could be due to missing variables, for example if there was another marketing channel that wasn't included, or simply due to noise in the data.

3. **At the top half, you can see Prob (F-statistic): 1.58e-96**. This probability value (p-value) represents the likelihood that there are **no** good predictors of the target variable — in this case, there are no good predictors of sales. Since the p-value is close to zero, we know that there is **at least** one predictor in the model that is a good predictor of sales.