1. Introduction

1.1 Motivation and Overview

A retail company “ABC Private Limited” wants to understand the customer purchase behaviour (specifically, purchase amount) against various products of different categories. They have shared purchase summary of various customers for selected high volume products from last month.
The data set also contains customer demographics (age, gender, marital status, city_type, stay_in_current_city), product details (product_id and product category) and Total purchase_amount from last month.
Now, they want to build a model to predict the purchase amount of customer against various products which will help them to create personalized offer for customers against different products.

The dataset consists of 550068 rows & 12 columns:

1. User_ID
2. Product_ID
3. Gender
4. Age
5. Occupation
6. City_Category
7. Stay_In_Current_City_Years
8. Marital_Status
9. Product_Category_1
10. Product_Category_2
11. Product_Category_3
12. Purchase (Response Variable)

1.2 Cross Industry Standard Process for Data Mining (CRISP-DM)

CRISP-DM is a data mining process model that describes commonly used approaches that data mining experts use to tackle problems.

Life cycle:

First phase - Business Understanding: Define the problem, set objectives, project planning.

Second phase - Data Understanding: Describing, exploring & verifying data quality.

Third phase - Data Preparation: Cleaning the data & Feature selection.

Fourth phase - Modeling: Fitting the data into a Machine Learning model for predictions.

Fifth phase - Evaluation: Evaluation of the results by applying the model on unseen data & calculating evaluation reports.

Sixth phase - Deployment: Conclusion, final report & presentation.

Phase I – Business Understanding

PROBLEM STATEMENT and OBJECTIVE:

A retail company “ABC Private Limited” wants to understand the customer purchase behaviour (specifically, purchase amount) against various products of different categories. They have shared purchase summary of various customers for selected high-volume products from last month.

The data set also contains customer demographics (age, gender, marital status, city_type, stay_in_current_city), product details (product_id and product category) and Total purchase_amount from last month.

Now, they want to build a model to predict the purchase amount of customer against various products which will help them to create personalized offer for customers against different products.

Phase II – Data Understanding

The dataset consists of 550068 rows & 12 columns:

- **User_ID:** Numerical value to uniquely identify a user.
- **Product_ID:** Alpha-numerical value to uniquely identify a product.
- **Gender**: Sex of user.
- **Age:** Age in bins.
- **Occupation:** Occupation (Masked)
- **City_Category:** Category of the City (A, B, C)
- **Stay_In_Current_City_Years:** Number of years stay in current city.
- **Marital_Status:** Marital Status (Masked)
- **Product_Category_1:** Product Category (Masked)
- **Product_Category_2:** Product may belong to another category also (Masked)
- **Product_Category_2:** Product may belong to another category also (Masked)
- **Purchase:** Purchase Amount (Target Variable)

Phase III – Data Preparation

Data Preparation mainly deals with Data Cleaning & Data Wrangling. The Dataset had null values in Product_Category_2 and Product_Category_3 columns which are needed to be treated as the model does not work well with null values. The Categorical variables also needed to be converted to numeric datatype as only numeric data can be passed in the model.

Feature engineering is the process of using **domain knowledge** of the data to create features that make machine learning algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process.

Feature engineering is the most important art in machine learning which creates the huge difference between a good model and a bad model.

For Feature Engineering, a method called Backward Elimination is used in this project.

In backward elimination, we start with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features

Phase IV – Modeling

It is also known as Algorithm selection for Predicting the best results.

Supervised learning is a type of system in which both input and desired output data are provided. Input and output data are labelled for classification to provide a learning basis for future data processing. Supervised learning problems can be further grouped into Regression and Classification problems.

- A regression problem is when the output variable is a real or continuous value, such as "salary" or "weight".
- A classification problem is when the output variable is a category like filtering emails "spam" or "not spam"

In this Project, Multiple Linear Regression, Decision Tree Regression and Random Forest Model is used to predict the Purchase Amount.

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

The **Random Forest** model is a type of additive model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:

$$g(x)=f0(x)+f1(x)+f2(x)+...$$

where the final model g is the sum of simple base models fi. Here, each base classifier is a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is called model ensembling. In random forests, all the base models are constructed independently using a different subsample of the data.

Phase V – Evaluation

The evaluation metrics that are available for Regression model are:

RMSE: Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

Where y hat is the predicted value and y is the original value.

Phase VI – Deployment

Deployment is the process of using your new insights to make improvements within your organization.

In general, the deployment phase of CRISP-DM includes two types of activities:

- Planning and monitoring the deployment of results
- Completing wrap-up tasks such as producing a final report and conducting a project review

2. Software Requirement Analysis

2.1 Problem Statement

A retail company “ABC Private Limited” wants to understand the customer purchase behaviour (specifically, purchase amount) against various products of different categories. They have shared purchase summary of various customers for selected high-volume products from last month.

The data set also contains customer demographics (age, gender, marital status, city_type, stay_in_current_city), product details (product_id and product category) and Total purchase_amount from last month.

Now, they want to build a model to predict the purchase amount of customer against various products which will help them to create personalized offer for customers against different products.

2.2 Modules and their functionalities

I. Scikit-Learn

Scikit-learn is probably the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

Figure 1

Components of scikit-learn:

Scikit-learn comes loaded with a lot of features. Here are a few of them to help you understand the spread:

Supervised learning algorithms: Think of any supervised learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g. Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of algorithms is one of the big reasons for high usage of scikit-learn. I started using scikit to solve supervised learning problems and would recommend that to people new to scikit / machine learning as well.

II. Multiple Linear Regression

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

In essence, multiple regression is the extension of ordinary least-squares (OLS) regression that involves more than one explanatory variable..

The Formula for Multiple Linear Regression Is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon$$

where, for $i = n$ observations:

y_i = dependent variable

x_i = expanatory variables

β_0 = y-intercept (constant term)

β_p = slope coefficients for each explanatory variable

ϵ = the model's error term (also known as the residuals)

Figure 2

III. Pandas

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

pandas is a NumFOCUS sponsored project. This will help ensure the success of development of pandas as a world-class open-source project, and makes it possible to donate to the project.

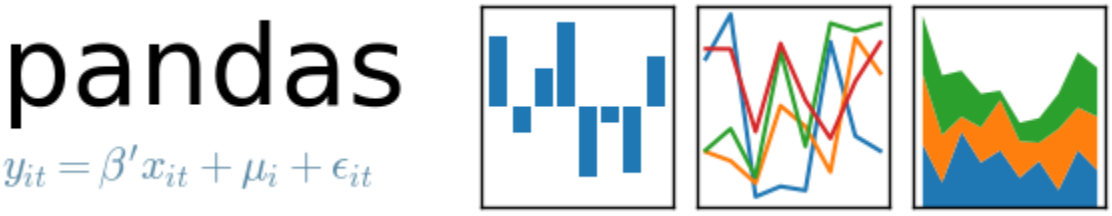

Figure 3

IV. statsmodel

statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics is available for each estimator. The results are tested against existing statistical packages to ensure that they are correct. The package is released under the open source Modified BSD (3-clause) license. The online documentation is hosted at statsmodels.org.

Figure 4

V. Cross Validation

Cross Validation is a technique which involves reserving a particular sample of a dataset on which you do not train the model. Later, you test your model on this sample before finalizing it.

K-Fold Cross Validation is a common type of cross validation that is widely used in machine learning. K-fold cross validation is performed as per the following steps:

Partition the original training data set into k equal subsets. Each subset is called a fold. Let the folds be named as f1, f2, …, fk.

For i = 1 to i = k

Keep the fold fi as Validation set and keep all the remaining k-1 folds in the Cross-validation training set.

Train your machine learning model using the cross-validation training set and calculate the accuracy of your model by validating the predicted results against the validation set.

Estimate the accuracy of your machine learning model by averaging the accuracies derived in all the k cases of cross validation.

In the k-fold cross validation method, all the entries in the original training data set are used for both training as well as validation. Also, each entry is used for validation just once.

Generally, the value of k is taken to be 10, but it is not a strict rule, and k can take any value.

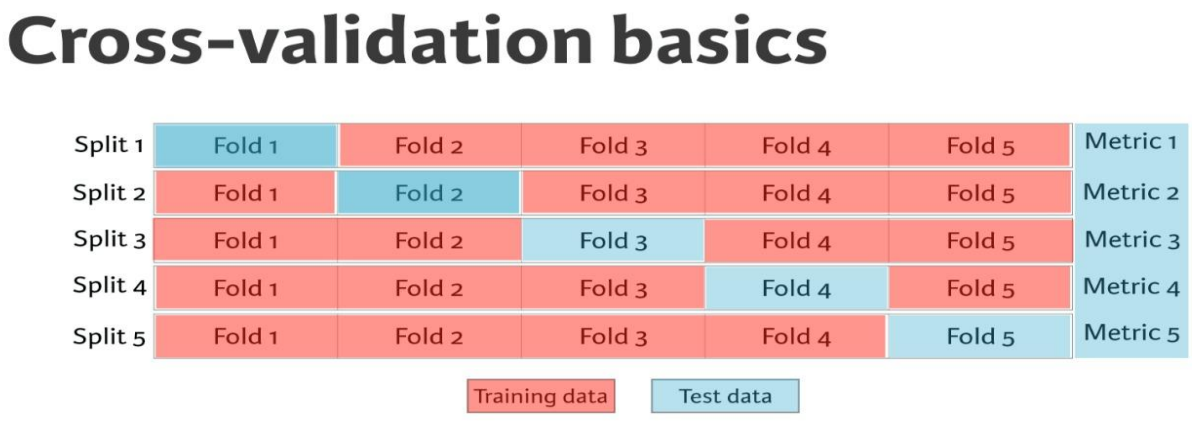

Figure 5

VI. Random Forest

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks. In this post, you are going to learn, how the random forest algorithm works and several other important things about it.

Difference between Decision Trees and Random Forests:

If you input a training dataset with features and labels into a decision tree, it will formulate some set of rules, which will be used to make the predictions.

Another difference is that deep decision trees might suffer from overfitting. Random Forest prevents overfitting most of the time, by creating random subsets of the features and building smaller trees using these subsets. Afterwards, it combines the subtrees. Note that this doesn't work every time and that it also makes the computation slower, depending on how many trees your random forest builds.

Important Hyperparameters:

The Hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster. I will here talk about the hyperparameters of sklearns built-in random forest function.

1. Increasing the Predictive Power

Firstly, there is the **n_estimators** hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.

Another important hyperparameter is **max_features**, which is the maximum number of features Random Forest considers to split a node. Sklearn provides several options, described in their documentation.

The last important hyper-parameter we will talk about in terms of speed, is **min_sample_leaf** . This determines, like its name already says, the minimum number of leafs that are required to split an internal node.

2. Increasing the Models Speed

The **n_jobs** hyperparameter tells the engine how many processors it is allowed to use. If it has a value of 1, it can only use one processor. A value of "-1" means that there is no limit.

random_state makes the model's output replicable. The model will always produce the same results when it has a definite value of random_state and if it has been given the same hyperparameters and the same training data.

Lastly, there is the **oob_score** (also called oob sampling), which is a random forest cross validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance. These samples are called the out of bag samples. It is very similar to the leave-one-out cross-validation method, but almost no additional computational burden goes along with it.

3. Implementation

Pre – requisites:

It includes importing of libraries and dataset for the preparation and modelling part to be executed later.

Let's start by importing some libraries and our data.

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:
```
train = pd.read_csv("train.csv")
test = pd.read_csv("test-comb.csv")
train.head()
```

Out[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	P
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	

Figure 6

Cleaning the dataset

2. Data Preprocessing

```
In [42]: train.isnull().sum()

Out[42]: User_ID                        0
         Product_ID                     0
         Gender                         0
         Age                            0
         Occupation                     0
         City_Category                  0
         Stay_In_Current_City_Years     0
         Marital_Status                 0
         Product_Category_1             0
         Product_Category_2        173638
         Product_Category_3        383247
         Purchase                       0
         dtype: int64

In [43]: test.isnull().sum()

Out[43]: Unnamed: 0                     0
         User_ID                        0
         Product_ID                     0
         Gender                         0
         Age                            0
         Occupation                     0
         City_Category                  0
         Stay_In_Current_City_Years     0
         Marital_Status                 0
         Product_Category_1             0
         Product_Category_2         72344
         Product_Category_3        162562
         Comb                           0
         dtype: int64
```

Figure 7

3. Feature Engineering

In [56]:
```
from sklearn.preprocessing import LabelEncoder
```

In [57]:
```
le = LabelEncoder()
```

In [58]:
```
train['Gender'] = le.fit_transform(train['Gender'])
test['Gender'] = le.fit_transform(test['Gender'])
```

In [59]:
```
train.head()
```

Out[59]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Purchase
0	0	0-17	10	A	2	0	3	8370
1	0	0-17	10	A	2	0	1	15200
2	0	0-17	10	A	2	0	12	1422
3	0	0-17	10	A	2	0	12	1057
4	1	55+	16	C	4+	0	8	7969

In [60]:
```
test.head()
```

Out[60]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1
0	1	46-50	7	B	2	0	1
1	1	26-35	17	C	0	0	3
2	0	36-45	1	B	4+	0	5
3	0	36-45	1	B	4+	0	4
4	0	26-35	1	C	1	0	4

In [61]:
```
train['Age'] = le.fit_transform(train['Age'])
test['Age'] = le.fit_transform(test['Age'])
```

In [62]:
```
train['City_Category'] = le.fit_transform(train['City_Category'])
```

In [63]:
```
test['City_Category'] = le.fit_transform(test['City_Category'])
```

Figure 8

Exploratory Data Analysis

In [23]:
```
#correlation matrix
f, ax = plt.subplots(figsize=(20, 9))
sns.heatmap(corr, vmax=.8,annot_kws={'size': 20}, annot=True);
```

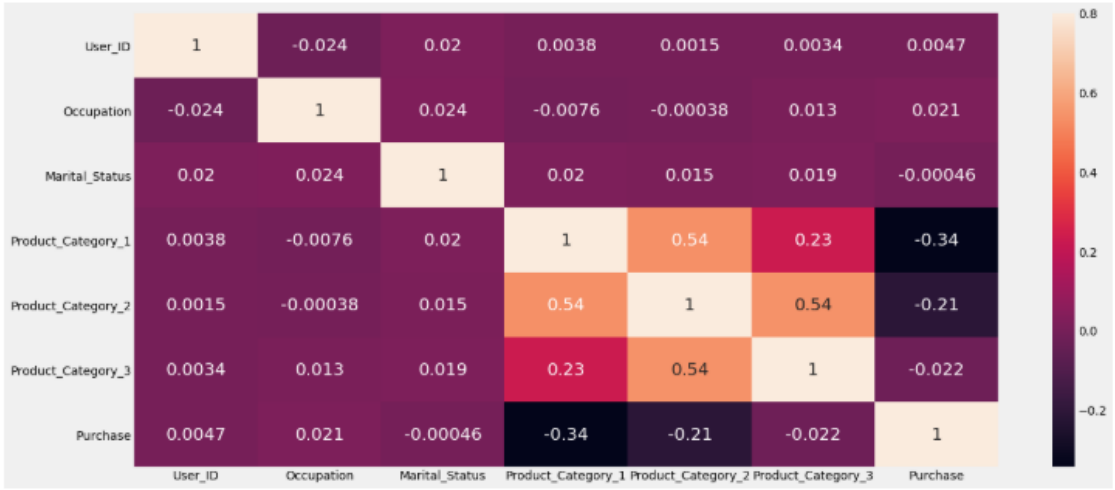

Figure 9

```
Out[8]: Text(0.5, 1.0, 'Purchase amount Distribution')
```

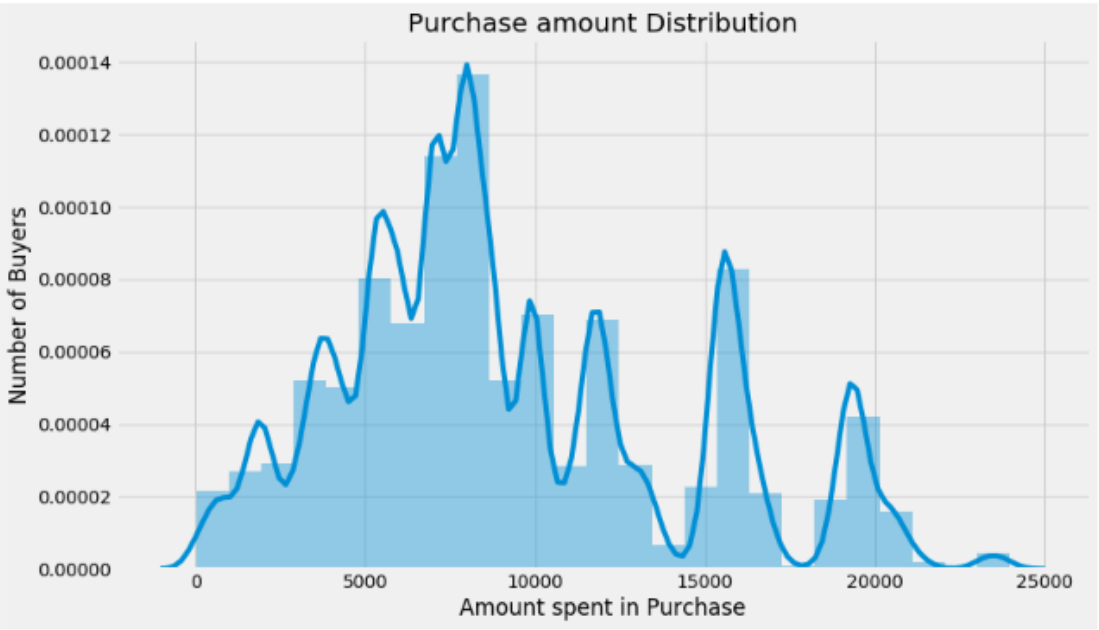

Figure 10

```
In [12]: sns.countplot(train.Occupation)
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1dfedfb14e0>
```

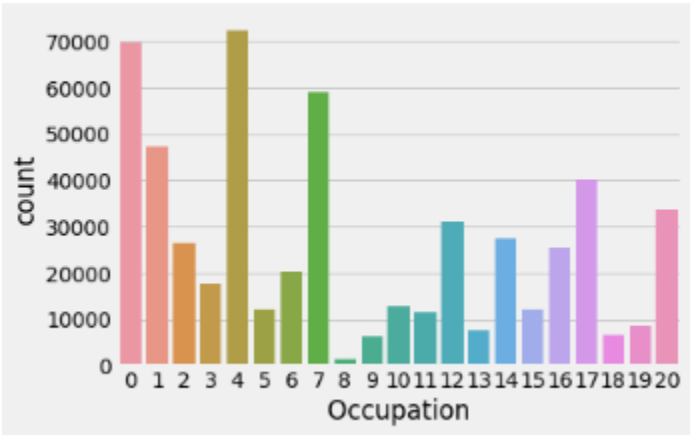

Figure 11

Fitting the model: Multiple Linear Regression

4.1 Multiple Linear Regression

In [75]:
```
X_train = train.drop('Purchase',axis=1)
y_train = train['Purchase']
X_test = test
```

In [76]:
```
from sklearn.linear_model import LinearRegression
```

In [77]:
```
reg = LinearRegression()
```

In [78]:
```
reg.fit(X_train, y_train)
```

Out[78]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [89]:
```
prediction1 = reg.predict(X_test)
```

In [90]:
```
prediction1
```

Out[90]:
```
array([11492.07110442, 10785.26962272,  9091.44149746, ...,
       10815.25902604,  7372.75994218,  9638.41030116])
```

In [91]:
```
pred = pd.DataFrame(prediction1, columns = ['Purchase'])
pred.head()
```

Out[91]:

	Purchase
0	11492.071104
1	10785.269623
2	9091.441497
3	9528.730516
4	9739.615697

In [94]:
```
final = pd.concat([train_join_test,pred],axis=1)
final.head()
```

Out[94]:

	User_ID	Product_ID	Purchase
0	1000001	P00069042	11492.071104
1	1000001	P00248942	10785.269623
2	1000001	P00087842	9091.441497
3	1000001	P00085442	9528.730516
4	1000002	P00285442	9739.615697

In [95]:
```
final.to_csv('finalpurchase-MLR.csv', index = False) # RMSE: 4713.9227352998 , Rank: 1344
```

Figure 12

Fitting the model: Decision Tree Regression

4.2 Decision Tree Regression

In [96]:
```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train, y_train)
```

Out[96]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=0, splitter='best')
```

In [97]:
```
prediction3 = regressor.predict(X_test)
pred = pd.DataFrame(prediction3, columns = ['Purchase'])
pred.head()
```

Out[97]:

	Purchase
0	16599.531250
1	10205.857143
2	6984.247191
3	2372.000000
4	2184.375000

In [99]:
```
final = pd.concat([train_join_test,pred],axis=1)
final.head()
```

Out[99]:

	User_ID	Product_ID	Purchase
0	1000001	P00069042	16599.531250
1	1000001	P00248942	10205.857143
2	1000001	P00087842	6984.247191
3	1000001	P00085442	2372.000000
4	1000002	P00285442	2184.375000

In [100]:
```
final.to_csv('finalpurchase-DTR.csv', index = False) # RMSE:  , Rank:
```

Figure 13

Fitting the model: Random Forest Regression

4.3 Random Forest Regression

In [101]:
```
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor(n_estimators = 10, random_state = 0)
reg.fit(X_train, y_train)
```

Out[101]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [102]:
```
prediction4 = regressor.predict(X_test)
pred = pd.DataFrame(prediction4, columns = ['Purchase'])
pred.head()
```

Out[102]:

	Purchase
0	16599.531250
1	10205.857143
2	6984.247191
3	2372.000000
4	2184.375000

In [104]:
```
final = pd.concat([train_join_test,pred],axis=1)
final.head()
```

Out[104]:

	User_ID	Product_ID	Purchase
0	1000001	P00069042	16599.531250
1	1000001	P00248942	10205.857143
2	1000001	P00087842	6984.247191
3	1000001	P00085442	2372.000000
4	1000002	P00285442	2184.375000

In [105]:
```
final.to_csv('finalpurchase-RFR.csv', index = False) # RMSE: 3119.9813685216  , Rank: 1143
```

Figure 14

Evaluation

Evaluation Results

Multiple Linear Regression- RMSE: 4713.9227352998 , Rank: 1344

Random Forest Regression- RMSE: 3119.9813685216 , Rank: 1143

Figure 15

References/Bibliography

http://scikit-learn.org/stable/documentation.html

https://en.wikipedia.org/wiki/Linear_regression

https://en.wikipedia.org/wiki/Random_forest

https://en.wikipedia.org/wiki/Cross-validation_(statistics)

https://pandas.pydata.org/pandas-docs/stable/

https://www.statsmodels.org/

https://seaborn.pydata.org/