Boeing Coding Challenge Write-Up

Prachatorn Joemjumroon

July 7, 2022

1 Write-Up

When working with the data provided, the first step I took was "preprocessing" the data given. This means that I was making sure I could use the data efficiently to use models on them and see if there are any abnormality that would be a problem for my program to work. I realised that this is supervised learning, as I have both the independent and dependent variables in the training dataset. When looking through the csv files, I noticed that some rows have missing data in them that would cause some sort of error within my program. My initial thought was to delete the rows with missing entries in them, but that proved to be the wrong approach as I noticed that the test dataset also have missing entries, and I could not delete rows in that dataset. My next approach was to fill in the missing entries, with zeros in the numerical columns, and "DNE"s in the categorical columns. I split the dependent variable columns, which were the Vehicle Trim and Selling Price, and the independent variables, which were the rest of the columns. Then, I converted all the columns with categorical variables into "dummy" variables, where the columns split into many with binary numbers, with 1 being true and 0 being false.

Since both the price and the trim of the vehicles are two different types of dependent variables, with the price being numerical and the trim being categorical, I decided to use two different types of models for them. I first predict the price of the vehicles as this was easier to make a model to solve. With all my models, I used the sklearn package as this allows me to make different types of models and is one that I am most comfortable using with. My first attempt I used the Linear Regression model, but that seemed to be not working as well as I had hoped, as I was getting astronomically higher numbers being predicted than what is expected with the training dataset. Then, with my second attempt, I used the Random Forest Regression model, which proved to be more reliable, as even though I am getting a lower r-squared score than the Linear Regression model, most of the predicted values I was getting was within the acceptable range of the training dataset.

I moved on to make a model for the vehicle trim, still using the sklearn package as with the model for the prices. Since the vehicle trim is categorical instead of numerical, I chose to use the Logistic Regression model. More specifically,

a multinomial Logistic Regression as there are more than two types of trim a vehicle can have. Within the code, there is a "solver" tag within the Logistic Regression function, which are algorithms that solves optimization problems, and I choose to use "liblinear" as the condtions for it is a dataset with high dimensions, which these datasets does have if you include all the dummy variables created. Finally, I combined and created a new csv file with both the predicted price and trim, along with the ListingID from the training dataset.

2 Code

2.1 models-price.py (Run this script first)

```
import pandas as pd

from sklearn.ensemble import RandomForestRegressor

# Read in both the training and test dataset.
train_data = pd.read_csv("training_DataSet.csv")
test_data = pd.read_csv("test_Dataset.csv")

# Filling the missing entries with 0 for numerical variables and
                                    DNE for categorical variables.
train_data[["SellerRating", "SellerRevCnt", "SellerZip", "
                                    VehListdays", "VehMileage", "
                                    VehYear", "Dealer_Listing_Price"]
                                    ] = train_data[["SellerRating", "
                                    SellerRevCnt", "SellerZip", "
                                    VehListdays", "VehMileage", "
                                    VehYear", "Dealer_Listing_Price"]
                                    ].fillna(0)
test_data[["SellerRating", "SellerRevCnt", "SellerZip", "
                                    VehListdays", "VehMileage", "
                                    VehYear"]]  = test_data[["
                                    SellerRating", "SellerRevCnt", "
                                    SellerZip", "VehListdays", "
                                    VehMileage", "VehYear"]].fillna(0
                                    )
train_data = train_data.fillna("DNE")
test_data = test_data.fillna("DNE")

# Seperating the independent and dependent variables in the
                                    training dataset.
X_train = train_data.drop("Dealer_Listing_Price", axis=1)
X_train = X_train.drop("Vehicle_Trim", axis=1)
y_train_price = train_data["Dealer_Listing_Price"]

# Combining both the training and test independent variables to
                                    make dummy variables at the same
                                    time.
car_data = [X_train, test_data]
total_data = pd.concat(car_data)

# Creating dummies variables in both the training and test dataset
                                    for categorical variables.
```

```
total_data_new = pd.get_dummies(total_data, columns=["SellerCity",
                                "SellerIsPriv", "SellerListSrc",
                                "SellerName", "SellerState", "
                                VehBodystyle", "VehCertified", "
                                VehColorExt", "VehColorInt", "
                                VehDriveTrain", "VehEngine", "
                                VehFeats", "VehFuel", "VehHistory
                                ", "VehMake", "VehModel", "
                                VehPriceLabel", "VehSellerNotes",
                                 "VehType", "VehTransmission"])


# Separating the training and testing dataset.
train_data_new = total_data_new.iloc[:6298,:]
test_data_new = total_data_new.iloc[6298:, :]

# Predicting the values of the prices of cars using the Random
                                Forest Regressor model.
rf = RandomForestRegressor(max_depth=2, random_state=42)
rf.fit(train_data_new, y_train_price)

y_rf_train_pred = rf.predict(train_data_new)
y_rf_test_pred = rf.predict(test_data_new)

# Making the new dataset with the ListingID from the test dataset
                                with predicted prices.
data = {"ListingID": test_data["ListingID"], "price_pred":
                                y_rf_test_pred}
df = pd.DataFrame(data)
df.to_csv("test_pred.csv")
```

2.2 models-trim.py

```
import pandas as pd

from sklearn.linear_model import LogisticRegression

# Read in both the training and test dataset.
train_data = pd.read_csv("training_DataSet.csv")
test_data = pd.read_csv("test_Dataset.csv")
test_pred = pd.read_csv("test_pred.csv")

# Removing rows where there are no values in a column.
train_data[["SellerRating", "SellerRevCnt", "SellerZip", "
                                VehListdays", "VehMileage", "
                                VehYear", "Dealer_Listing_Price"]
                                ] = train_data[["SellerRating", "
                                SellerRevCnt", "SellerZip", "
                                VehListdays", "VehMileage", "
                                VehYear", "Dealer_Listing_Price"]
                                ].fillna(0)
test_data[["SellerRating", "SellerRevCnt", "SellerZip", "
                                VehListdays", "VehMileage", "
                                VehYear"]]  = test_data[["
```

```
                                    SellerRating", "SellerRevCnt", "
                                    SellerZip", "VehListdays", "
                                    VehMileage", "VehYear"]].fillna(0
                                    )
train_data = train_data.fillna("DNE")
test_data = test_data.fillna("DNE")

# Seperating the independent and dependent variables in the
                                    training dataset.
X_train = train_data.drop("Dealer_Listing_Price", axis=1)
X_train = X_train.drop("Vehicle_Trim", axis=1)
y_train_trim = train_data["Vehicle_Trim"]

# Combining both the training and test independent variables to
                                    make dummy variables at the same
                                    time.
car_data = [X_train, test_data]
total_data = pd.concat(car_data)

# Creating dummies variables in both the training and test dataset
                                    for categorical variables.
total_data_new = pd.get_dummies(total_data, columns=["SellerCity",
                                    "SellerIsPriv", "SellerListSrc",
                                    "SellerName", "SellerState", "
                                    VehBodystyle", "VehCertified", "
                                    VehColorExt", "VehColorInt", "
                                    VehDriveTrain", "VehEngine", "
                                    VehFeats", "VehFuel", "VehHistory
                                    ", "VehMake", "VehModel", "
                                    VehPriceLabel", "VehSellerNotes",
                                     "VehType", "VehTransmission"])

# Separating the training and testing dataset.
train_data_new = total_data_new.iloc[:6298,:]
test_data_new = total_data_new.iloc[6298:, :]

# Predicting the values of the trims of cars using the Multinominal
                                     Logistic Regressor model.
logmodel = LogisticRegression(solver="liblinear")
logmodel.fit(train_data_new, y_train_trim)
y_trim_pred = logmodel.predict(test_data_new)

# Adding on the prediction to the trim onto the predicted values
                                    csv file.
data = {"ListingID": test_pred["ListingID"], "trim_pred":
                                    y_trim_pred, "price_pred":
                                    test_pred["price_pred"]}
df = pd.DataFrame(data)
df.to_csv("test_pred.csv")
```