Task 2:

Data Analysis Report

Keniyah Chestnut

Data Science Capstone — D606

012601305

A. Research Question

The goal of this project is to determine what vehicle features most influence the price of used cars and whether a Random Forest regression model can accurately predict those prices. This question is important in the context of the used car resale market, where pricing accuracy directly impacts sales and profitability. Dealerships and sellers often rely on subjective estimates or generic pricing tools, which can result in overpricing or undervaluing vehicles.

By building a data-driven predictive model, we aim to provide a more objective and consistent method of estimating car value. This supports more efficient pricing strategies and reduces reliance on manual judgment.

Hypothesis: There is a statistically significant relationship between features such as brand, model year, mileage, fuel type, engine type, and title status, and the market price of a used car.

Null Hypothesis: There is no significant relationship between the listed vehicle features and market value.

This analysis is important because accurate car pricing can help eliminate human error and bias from the resale process, improving decision-making for dealerships and buyers alike.

B. Data Collection

The dataset used in this project was provided by WGU and includes 4,010 records of used cars. Each record contains details such as brand, model, model year, mileage, fuel type, engine type, transmission, exterior and interior color, accident history, title status, and market price.

Advantage: Since the dataset was pre-collected and curated by WGU, it is clean and structured, which reduces the need for data wrangling and supports reproducible analysis.

Disadvantage: The origin and collection methods are unknown, which limits the ability to assess potential collection bias or missing data patterns.

Challenge: One specific challenge was handling missing values in fields like title_status and accident. Without documentation, it was unclear whether blanks represented clean titles or missing entries. To resolve this, I categorized all missing entries as “Unknown” to preserve potentially meaningful distinctions without discarding data.

C. Data Extraction and Preparation

All data cleaning and preparation was performed using Python in Jupyter Notebook. I used the pandas library for data inspection, cleaning, and transformation, and matplotlib and seaborn for

data visualization. Regular expressions were applied to clean the mileage column. One-hot encoding was performed using pd.get_dummies() to prepare categorical variables for modeling.

Justification: Python is widely used in data science and provides powerful libraries for preprocessing and modeling. It enables transparent, reproducible, and scriptable workflows that can be reviewed or automated.

Advantage: Python allows fine-grained control over how each column is processed.

Disadvantage: It requires careful handling of missing values and categorical variables, which can introduce errors if not managed properly.

The first step in preparing the data was identifying missing values. Some columns, such as the title status, had blank entries. In this case, every car either had a "clean" title or a missing value. It is possible that the missing data could indicate a pattern. For example, these vehicles may have rebuilt or branded titles that were not properly recorded.

```
# STEP 1: Load and clean data
df = pd.read_excel("Used Car Data.xlsx", skiprows=1)
df.columns = [
    "brand", "model", "model_year", "mileage", "fuel_type", "engine",
    "transmission", "ext_color", "int_color", "accident", "clean_title", "price"
]

#Looking formissing data
print(df.isnull().sum())
```

```
brand            0
model            0
model_year       0
mileage          0
fuel_type      170
engine           0
transmission     0
ext_color        0
int_color        0
accident       113
clean_title    596
price            0
dtype: int64
```

To address this, I replaced all missing values in the title column with the label “Unknown.” This approach allows the model to treat the absence of information as a distinct category, which could help in identifying patterns in car value. One advantage of this method is that it preserves potentially useful information about vehicles that may be less desirable due to unlisted title issues. However, a limitation is the lack of documentation explaining why the values are missing, which introduces some uncertainty. For instance, the missing entries may simply reflect clean titles that were never updated, and assuming otherwise could be misleading.

```
# Filling unknown data with 'Unknown'
df['fuel_type'].fillna("Unknown", inplace=True)
df['accident'].fillna("Unknown", inplace=True)
df['clean_title'].fillna("Unknown", inplace=True)

# Confirming all missing values were handled
print(df.isnull().sum())
```

```
brand           0
model           0
model_year      0
mileage         0
fuel_type       0
engine          0
transmission    0
ext_color       0
int_color       0
accident        0
clean_title     0
price           0
dtype: int64
```

Even with that uncertainty, treating missing values as a separate category gives the model more flexibility in learning from the available data.

After addressing missing values, I cleaned the mileage column, which contained non-numeric characters such as letters and punctuation. These values had to be removed so the mileage could be processed as a numeric feature in the analysis. I used regular expressions to extract only the numeric values and then converted the result to integers for modeling.

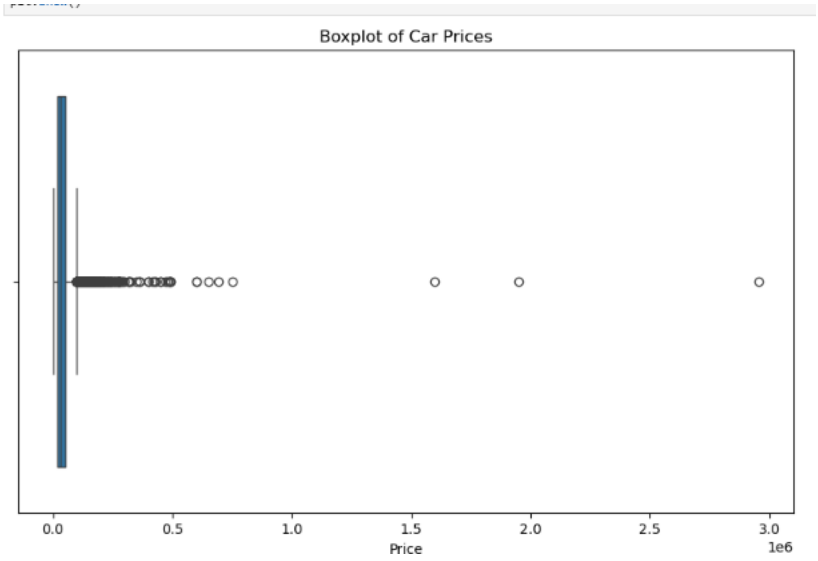
```
# Removing non-numeric characters from mileage and keeping only the numbers
df['mileage'] = df['mileage'].str.replace(r'[^0-9]', '', regex=True)

# Convert the column to integers for further analysis
df['mileage'] = df['mileage'].astype(int)

# Confirm result
print(max(df['mileage']))
```

```
405000
```

Next, I generated a boxplot of the car prices to identify outliers. The visualization revealed a small number of extremely high-priced vehicles that were far outside the typical price range. These values had the potential to skew the model, especially since most people do not purchase used cars for over $100,000.



To reduce this distortion, I removed vehicles with prices above $100,000 from the dataset. These cars represented less than 5 percent of the total data and were unlikely to be helpful in building a

model intended for general use. Removing these outliers allowed the model to focus on more common vehicle prices.

```
]: # Filter the DataFrame to include only rows where price is less than or equal to 100,000
df_filtered = df[df['price'] <= 100000]

# Check price distribution after filtering
print(df_filtered['price'].describe())
```

```
count      3775.000000
mean      33694.054570
std       21611.948944
min        2000.000000
25%       16500.000000
50%       29798.000000
75%       45880.000000
max      100000.000000
Name: price, dtype: float64
```

One advantage of this approach is that it prevents the model from being skewed by rare, luxury vehicles that do not represent typical transactions. A disadvantage is that the model may not be able to accurately predict prices for high-end vehicles that fall outside the normal range.

D:Analysis

Before fitting the model, I encoded the categorical variables so they could be used in the Random Forest algorithm. Since Random Forests require numerical input, I used one-hot encoding to convert each category into a separate binary column. This method is useful when dealing with non-ordinal variables, such as brand or fuel type, that don't follow a natural order.

```
# One-hot encoding for categorical variables with corrected column names
encoded_df = pd.get_dummies(df_filtered, columns=[
    'brand', 'fuel_type', 'transmission', 'ext_color',
    'int_color', 'accident', 'engine', 'model', 'clean_title'
])

# Identifying boolean columns and converting them to integers
boolean_columns = encoded_df.select_dtypes(include='bool').columns
encoded_df[boolean_columns] = encoded_df[boolean_columns].astype(int)

# Check the result
print(encoded_df)
```

```
      model_year  mileage  price  brand_Acura  brand_Alfa  brand_Aston  \
0           2013    51000  10300            0           0            0
1           2021    34742  38005            0           0            0
2           2022    22372  54598            0           0            0
3           2015    88900  15500            0           0            0
4           2021     9835  34999            0           0            0
...          ...      ...    ...          ...         ...          ...
4003        2018    53705  25900            0           0            0
4005        2022    10900  53900            0           0            0
4006        2022     2116  90998            0           0            0
4007        2020    33000  62999            0           0            0
4008        2020    43000  40000            0           0            0

      brand_Audi  brand_BMW  brand_Bentley  brand_Buick  ...  model_i3 94 Ah  \
0              0          0              0            0  ...               0
1              0          0              0            0  ...               0
2              0          0              0            0  ...               0
3              0          0              0            0  ...               0
4              1          0              0            0  ...               0
...          ...        ...            ...          ...  ...             ...
4003           0          0              0            0  ...               0
4005           1          0              0            0  ...               0
4006           0          0              0            0  ...               0
4007           0          0              0            0  ...               0
4008           0          1              0            0  ...               0

      model_i3 Base  model_i3 Base w/Range Extender  model_i8 Base  \
0                 0                               0              0
1                 0                               0              0
2                 0                               0              0
3                 0                               0              0
4                 0                               0              0
...             ...                             ...            ...
4003              0                               0              0
4005              0                               0              0
4006              0                               0              0
4007              0                               0              0
4008              0                               0              0

      model_tC Anniversary Edition  model_tC Base  \
0                                0              0
1                                0              0
2                                0              0
3                                0              0
4                                0              0
...                            ...            ...
```

Next, I proceeded to build and evaluate the predictive model using a Random Forest regression algorithm. This model is well-suited for tabular data with both numerical and categorical features and works by building multiple decision trees and averaging their results to improve prediction accuracy and reduce overfitting.

First, I split the dataset into training and test sets using a 60/40 ratio. Then, I initialized and trained the Random Forest model with 100 estimators. Finally, I used the test set to generate predictions and evaluate model performance using three common regression metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2).

```
# Splitting the data and training it
X = encoded_df.drop('price', axis=1)
y = encoded_df['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=2)

# Running the Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=2)

# Training the model and making predictions
rf_model.fit(X_train, y_train)
y_test_pred = rf_model.predict(X_test)

# Evaluating accuracy
print("Test Mean Absolute Error:", mean_absolute_error(y_test, y_test_pred))
print("Test Mean Squared Error:", mean_squared_error(y_test, y_test_pred))
print("Test R-squared:", r2_score(y_test, y_test_pred))
```

```
Test Mean Absolute Error: 8024.776523178808
Test Mean Squared Error: 132764065.27467822
Test R-squared: 0.69941072865162
```

These results indicate that the model predicts used car prices within an average error of approximately $8,025. The R^2 value of 0.699 suggests that nearly 70% of the variability in car prices is explained by the features included in the model.

One advantage of using Random Forest is its ability to handle a large number of features and interactions between them without requiring extensive parameter tuning. However, a disadvantage is that the model may become less interpretable and may require more memory and computational power, especially with many unique categories in features like brand or engine type.

E. Data Summary and Implications

As shown above, the model achieved a mean absolute error of approximately $8,024.78, which means it can estimate car prices within about $8,025 on average. The R-squared value of 0.6990 indicates that the model is able to explain about 70% of the variance in used car prices. These results directly address the research question and demonstrate that machine learning techniques like Random Forest regression can be used to predict used car values based on features such as brand, mileage, year, fuel type, and title status.

One limitation of this analysis is that the model may not generalize well to predict the prices of extremely high-value vehicles. Cars priced over $100,000 were excluded from the dataset due to their rarity and potential to skew the model. As a result, the model does not account for features or patterns unique to the luxury car segment.

Two directions for future study include:

1. Expanding the dataset: Increasing the number of records, especially for underrepresented brands and car types, would help the model generalize better. Some specifications only appeared once or twice in the dataset, which limited the model's ability to learn from them. A larger dataset would improve training and help the model recognize patterns more consistently.
2. Segmenting the data by vehicle price range: The dataset includes vehicles priced from around $2,000 to nearly $3 million. This wide range creates a challenge when trying to

predict typical used car values. A better approach may be to separate standard vehicles from high-end or luxury vehicles and train separate models for each category. This would allow the model to focus on patterns specific to the market segment it is designed to predict.

In summary, the current model is a good starting point for identifying what makes a used car more or less valuable. However, because most used cars in this dataset fall under $50,000, a model with a prediction error of over $8,000 is not yet accurate enough to provide pricing for every vehicle. Implementing the two improvements above would help increase precision and allow the model to scale for more specific use cases.

References

No sources were used besides official WGU materials.