Microsoft Malware Prediction

Pavel Zimin

Problem Statement

Computer infection by malware constitutes a serious security problem that could harm consumers and businesses in many different ways. The ability to predict the chances of malware infection before they occur would benefit consumers and businesses. The signal that the infection is likely to occur would allow for timely countermeasures to be applied against it. This project would benefit software manufacturers who would be able to incorporate the model into their software that would allow for additional security measures aimed at preventing the infection by malware.

The dataset that I will use for this project is provided by Microsoft and available as part of the kaggle competition (https://www.kaggle.com/c/microsoft-malware-prediction/data). These data include telemetry properties and infection records of Windows machines as produced by the Windows Defender software. Each row represents an individual machine. Each column represents a variable. The “HasDetections” columns of the train data indicates whether the machine is infected. The goal of the competition is to be able to predict the value of “HasDetections” column for the test data. The train data contains 8,921,483 rows and 83 columns. This is a reasonably large size dataset that allows to explore many Machine Learning techniques. The dataset was resampled such that the frequency of machines with malware approximately matches the frequency of machines without malware detected. One caveat hear is that the test set was collected later in time than the train set and therefore the accuracy metrics obtained from the train set may not necessarily match the accuracy metrics of the test set. However, the test set accuracy metrics will be a good measure of how the model would generalize to the future data. The evaluation metric for this competition was selected as area under the ROC curve between the predicted probability and the observed label.

To solve this problem, I will build models based on logistic regression, Random Forest, and neural networks. Since the dataset is large, I may need to do some feature selection.

At the conclusion of this project, I will produce a Jupyter notebook with the code, a report and presentation slides.

Description of the Dataset and Cleaning Steps

The dtype of each column was specified in order to reduce the memory usage when reading from disk. The train dataset contains 8,921,483 rows and 83 columns, while the test dataset contains 7,853,253 rows and 82 columns. The goal of the kaggle competition is to be able to predict the 'HasDetections' column that is missing from the test data.

To read the data into memory, the dtypes were specified to read_csv function such that the pandas DataFrame takes less RAM space. The dataset's attributes were classified manually into the categorical and numeric groups based on the attribute description on the kaggle web page. Inspecting the value counts of the categorical Missing values are either coded as NaN or using various placeholders, such as 'UNKNOWN', 'Unknown', 'unkn' or 'Unspecified'. These values were substituted with np.nan values. Features with more than 1/3rd values missing were deleted from further analysis. In addition, for each column with missing data I created another column in which missing values were coded as 1, and 0 otherwise. Further analysis of missing values using the `missingno` package revealed that there are observations with 0-20 values missing. The correlation heatmap of missing values shows that there are features that are highly likely to be missing similarly (Figure 1). For example, missing values of 'IsProtected' are highly correlated with the missing values of 'AVProductStatesIdentifier', 'AVProducstsInstalled' and 'AVProductsEnabled'. This makes sense since these attributes describe the parameters of antivirus software. Another group of attributes that are highly correlated for the missing values: 'Census_ProcessorManufacturerIdentifier', 'Census_ProcessorCoreCount',

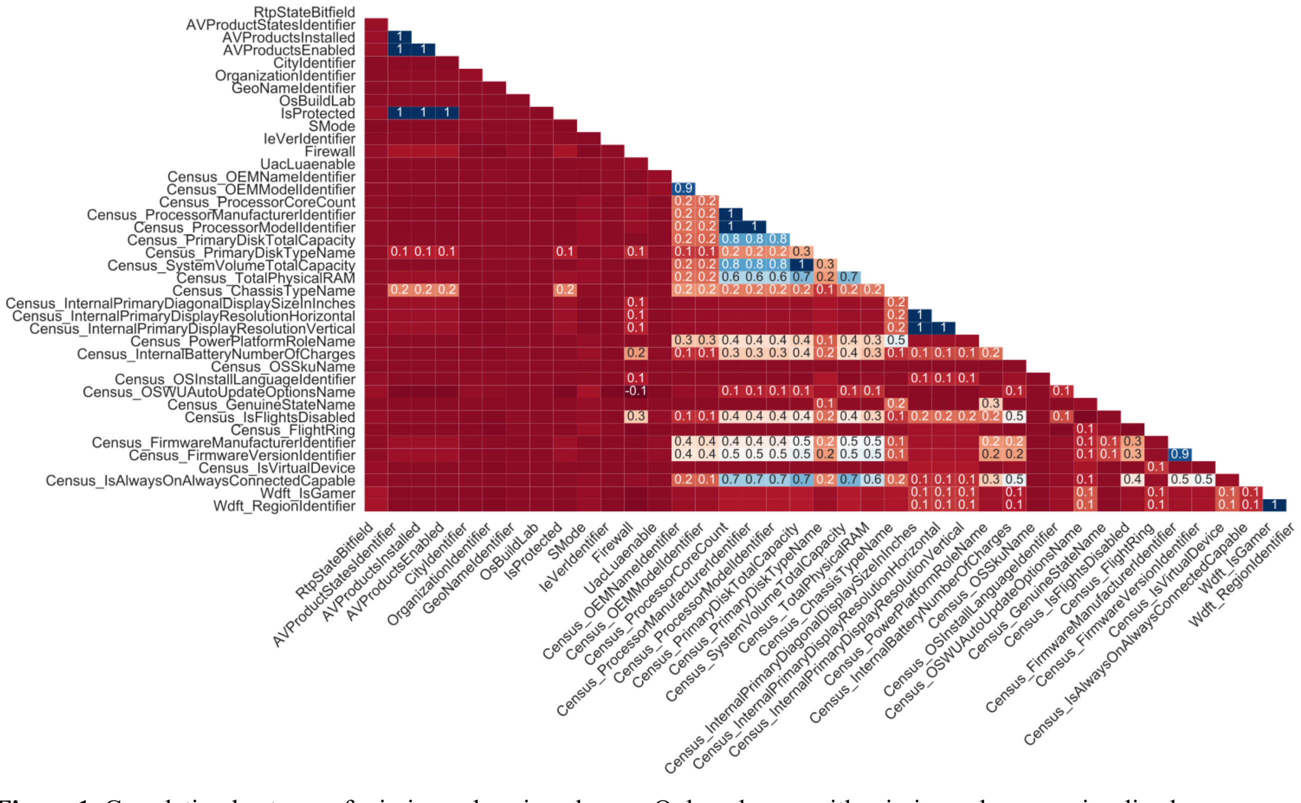


Figure 1. Correlation heatmap of missing values in columns. Only columns with missing values are visualized.

'Census_ProcessorModelIdentifier'. Again, this is not surprising since these features are the processor's parameters and will tend to be missing together.

Since most machine learning models require the complete dataset, I filled the categorical missing values with 0, while adding 1 to the existing values. The numeric columns were filled with the median values. The median values of all the numeric columns were saved for further use when need to fill the missing values of the test data or future data.

For outlier detection of numeric features 2 methods were used: 1) univariate outliers were identified as lower than 25th percentile minus 1.5 interquartile range or greater than 75th percentile plus 1.5 interquartile range, 2) multivariate outliers were detected by first normalizing the values by subtracting the mean and dividing by the standard deviation, then the principal component analysis (PCA) was applied (Figure 2). The scatterplot showing the distribution of the principal component 1 and principal component 2 shows the outlying data points with the greatest variance. Both methods were able to identify the outliers, however I decided against removing the outliers from further analysis since the most promising machine learning algorithms such as random forest are immune to outliers. In additions, outliers could contain information that could be predictive for the prediction.

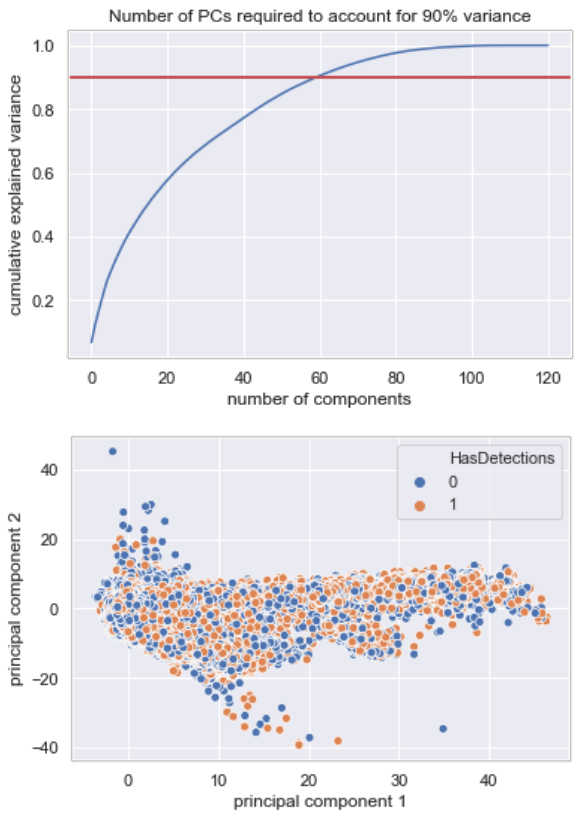
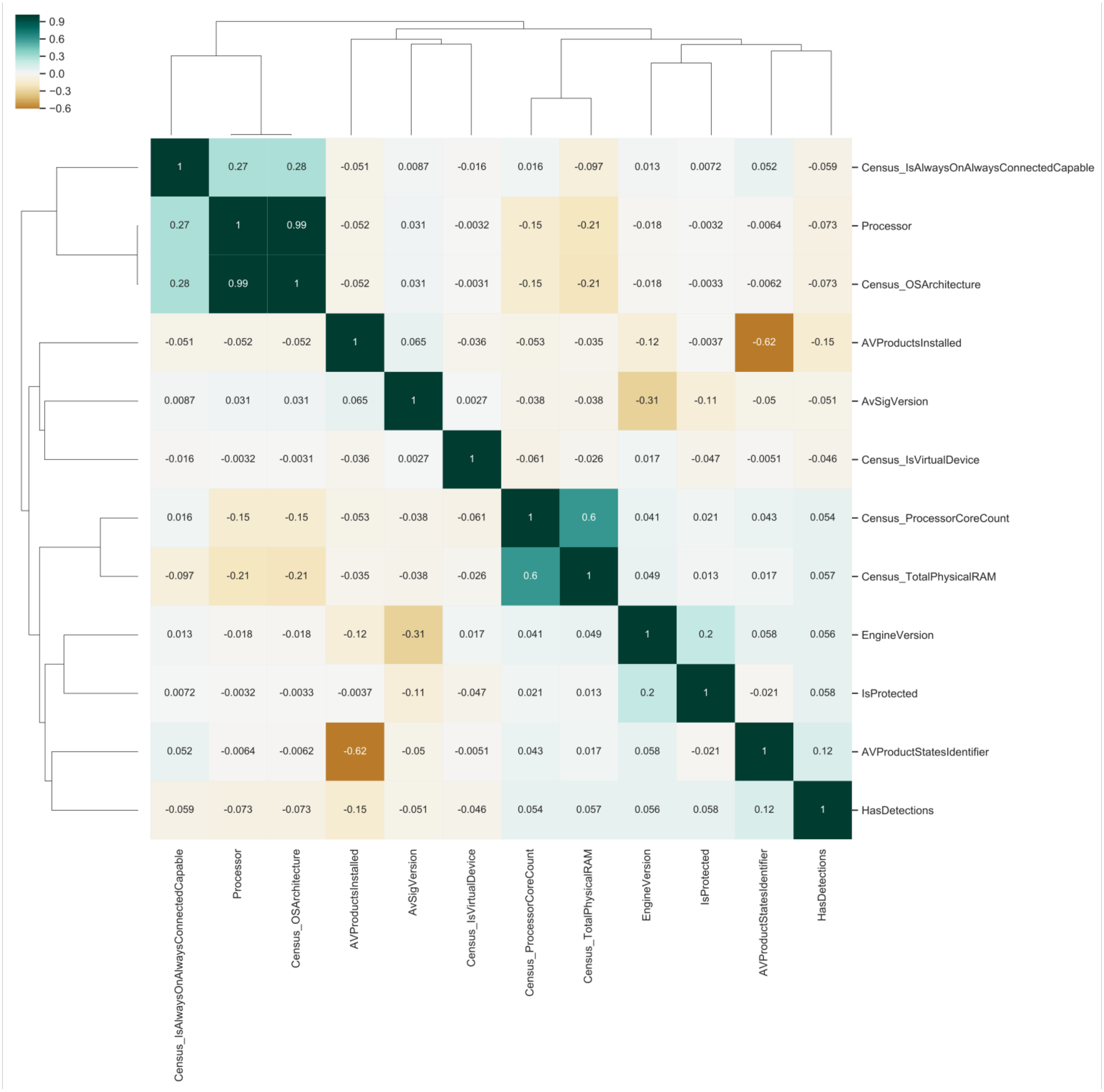


Figure 2. Principal Component Analysis of the training data. Top panel shows the cumulative explained variance as a function of the number of components. Bottom panel shows the scatter plot of the first 2 principal components.

Exploratory Data Analysis

What features are correlated with the target variable?

To identify the features that correlated with the target variable ‘HasDetectsions’, I plotted the hierarchical cluster heatmap (Figure 3). Due to the large number of features in the dataset for the figure I only filtered the features with the highest absolute correlation coefficient to the target variable. The highly correlated features are ‘AVProductsInstalled’, ‘AVProductStatesIdentifier’, ‘Processor’, ‘Census_OSArchitecture’.



	Census_IsAlwaysOnAlwaysConnectedCapable	Processor	Census_OSArchitecture	AVProductsInstalled	AvSigVersion	Census_IsVirtualDevice	Census_ProcessorCoreCount	Census_TotalPhysicalRAM	EngineVersion	IsProtected	AVProductStatesIdentifier	HasDetections
Census_IsAlwaysOnAlwaysConnectedCapable	1	0.27	0.28	-0.051	0.0087	-0.016	0.016	-0.097	0.013	0.0072	0.052	-0.059
Processor	0.27	1	0.99	-0.052	0.031	-0.0032	-0.15	-0.21	-0.018	-0.0032	-0.0064	-0.073
Census_OSArchitecture	0.28	0.99	1	-0.052	0.031	-0.0031	-0.15	-0.21	-0.018	-0.0033	-0.0062	-0.073
AVProductsInstalled	-0.051	-0.052	-0.052	1	0.065	-0.036	-0.053	-0.035	-0.12	-0.0037	-0.62	-0.15
AvSigVersion	0.0087	0.031	0.031	0.065	1	0.0027	-0.038	-0.038	-0.31	-0.11	-0.05	-0.051
Census_IsVirtualDevice	-0.016	-0.0032	-0.0031	-0.036	0.0027	1	-0.061	-0.026	0.017	-0.047	-0.0051	-0.046
Census_ProcessorCoreCount	0.016	-0.15	-0.15	-0.053	-0.038	-0.061	1	0.6	0.041	0.021	0.043	0.054
Census_TotalPhysicalRAM	-0.097	-0.21	-0.21	-0.035	-0.038	-0.026	0.6	1	0.049	0.013	0.017	0.057
EngineVersion	0.013	-0.018	-0.018	-0.12	-0.31	0.017	0.041	0.049	1	0.2	0.058	0.056
IsProtected	0.0072	-0.0032	-0.0033	-0.0037	-0.11	-0.047	0.021	0.013	0.2	1	-0.021	0.058
AVProductStatesIdentifier	0.052	-0.0064	-0.0062	-0.62	-0.05	-0.0051	0.043	0.017	0.058	-0.021	1	0.12
HasDetections	-0.059	-0.073	-0.073	-0.15	-0.051	-0.046	0.054	0.057	0.056	0.058	0.12	1

Figure 3. Correlation clustermap of the dataset features. Only features with the highest correlation with the target variable are visualized.

Next, I plotted the fraction of the positive target variable as a function of selected categorical features. I also built a weighted least square models using the value count as a weight. Data points with the large count were weighted larger than the points with the fewer count. This

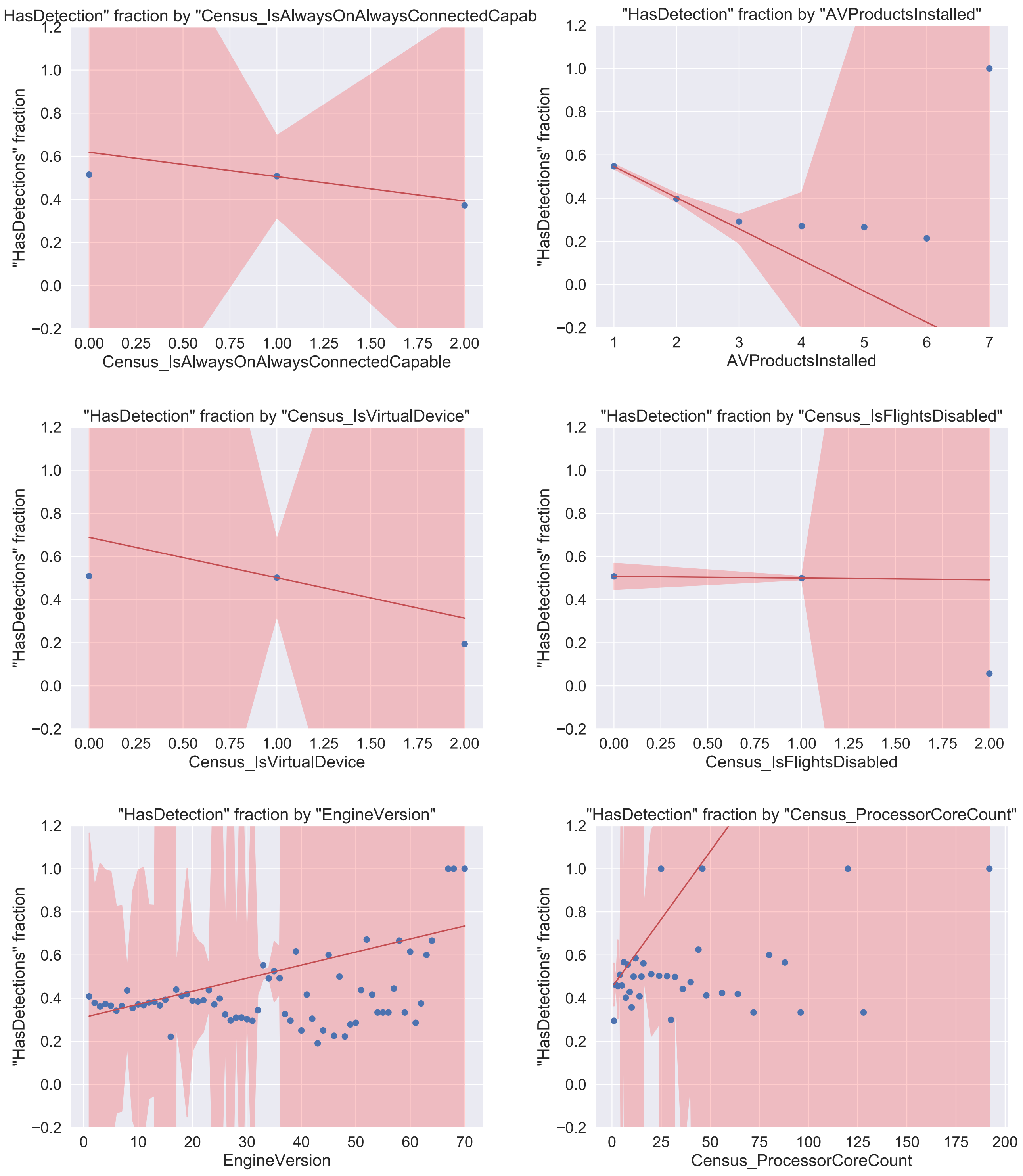


Figure 4. Fraction of the positive target variable as a function of selected categorical feature values. Weighted least square models were created using the value count for each categorical value as weight. Prediction interval is shown in red shade.

allows to minimize the effect of the points with the few counts and enable to see the model that reflects the large portion of data. Below is the analysis of the graphs shown in Figure 4:

1. \`Census_IsAlwaysOnAlwaysConnectedCapable\`. This variable has the majority of data points with the value of 1. Since there are very few data points in either direction from 1, the feature's coefficient is not significantly different from 0 (p value is 0.25).
2. \`AVProductsInstalled\`. The target variable positive fraction shows significant dependence at the low values of \`AVProductsInstalled\` variable (slope of -0.1443, p value = 0.000). The fraction of the target variable at larger values of \`AVProductsInstalled\` variable cannot be predicted with this model due to the few value counts.
3. \`Census_IsVirtualDevice\` variable shows an issue similar to the \`Census_IsAlwaysOnAlwaysConnectedCapable\` variable. The most data points have a value of 1, few data points in either direction from 1. The feature's coefficient is not significantly different from 0 (p value = 0.424).
4. \`Census_IsFlightsDisabled\` variable is approximately equally split by the target variable at values 0, and 1. At value 2, the target variable positive fraction is below 0.5, but there are a few point counts, hence nonsignificant slope (p value = 0.261).
5. \`EngineVersion\` has a larger spread of possible values, but the data are concentrated at a few points. The target variable positive fraction tends to increase with the increase in the \`EngineVersion\` values. The feature's slope from the model is 0.0061 (p value = 0.000).
6. \`Census_ProcessorCoreCount\` has also a large spread of possible values, but the data is concentrated at the lower values where the relationship of the positive fraction of the target variable is increasing with the increase in the values of \`Census_ProcessorCoreCount\` values (slope = 0.0126, p value = 0.000).

Graphs visualizing additional features can be found in the accompanying Jupyter notebook.

Overall, the EDA demonstrated that the correlation of the features to the target variable is quite low, with some features showing statistically significant relationships with the target variable.