Question 1

1. After reading the file and processing the data in Spark, a 25% random sample was taken out of the total dataset to try a cross-validation pipeline. This pipeline had three different models and a parameter grid that allowed to try several configurations given the time constraints.

   - Decision Tree Classifier (6 model combinations) :
     - maxDepth = 5, 10 ,15
     - maxBins = 16, 32
   - Decision Tree Regression (9 model combinations) :
     - maxDepth = 5, 10 ,15
     - maxBins = 6, 16, 32
   - Logistic Regression (8 model combinations) :
     - regParam = 0.01, 0.05
     - maxIter = 5, 10
     - maxIter = 0, 0.1

   The fact that the pipeline contained our 3 models in separate stages optimized the time spark would have taken if we had calculated each model separately, because spark used the same cached train/test split to evaluate each model.

   The best Decision Tree Classifier was the one with with maxDepth = 15 and maxBins = 32, which obtained an accuracy of 0.703.

   The best Decision Tree Regressor was the one with maxDepth = 15 and maxBins = 32, which obtained an accuracy of 0.704

   The best Logistic Regression was the one with regParam = 0.01, maxIter = 10 and elasticNetParam = 0, which obtained an accuracy of 0.635

   Now we will use these parameters to obtain the performance on the full dataset and compare the different algorithms.

2. Using the full dataset, with a 80/20 train-test split, we got the following results for our models.

   - Decision Tree Classifier (maxDepth=15, maxBins=32):
     - Accuracy = 0.718
     - AUC = 0.717
   - Decision Tree Regression (maxDepth=15, maxBins=32):
     - Accuracy = 0.717
     - AUC = 0.716
   - Logistic Regression (regParam=0.01, maxIter=10, elasticNetParam=0):

  - Accuracy = 0.635
  - AUC = 0.635

The winner by a little margin was the Decision Tree Classifier, followed closely by the Decision Tree Regressor, finally the Logistic Regression had much worse results.

The total training time for these 2 parts was around 11 minutes, where the difference between 10 and 20 cores was not very noticeable (around 30 seconds).

The most important features for each model were the following:

- Decision Tree Classifier (maxDepth=15, maxBins=32):
  - high_5 , 0.32
  - high_7 , 0.14
  - high_6 , 0.09
- Decision Tree Regression (maxDepth=15, maxBins=32):
  - high_5 , 0.32
  - high_7 , 0.14
  - high_6 , 0.09
- Logistic Regression (regParam=0.01, maxIter=10, elasticNetParam=0):
  - high_6 , 0.84
  - low_6 , 0.69
  - high_2 , 0.66

We can see that binarizing the results from the Decision Tree Regressor at the end, transforms this model into a Classifier with the same properties as the Decision Tree Classifier.

Question 2

Preprocessing

The first processing step in this dataset, was to remove any column that contained a null value, this transforms the original 13.1 million rows file into a 3.7 million rows dataset.

Then we dropped some features which were not informative like "Household ID" or "Vehicle".

We also dropped some categorical features (Blind_Model, Blind_submodel) which even though they could be informative, their dimensionality was too big , for example Blind_Model had 1300 different categories, and Blind_Submodel had 2739. Intuitively I think if there was an effect in the car model, this could be captured by the variable Blind_Make (the brand).

To be able to use the categorical features, we transformed them into One Hot Encodings, this incremented our number of features a lot.

Finally given that there were many features a PCA was done to select the top principal components that contained 90% of the variance.

Given that the data is highly unbalanced, we will evaluate the models using different options (full dataset, undersampling, oversampling, only positive values).

Linear Regression

A standard Linear Regression was fitted to the 4 possible datasets, and the results were the following:

	RMSE	R2	Train Size	Test Size
Full Dataset	35	0.0	2794326	931975
Only Positives	312	0.0	17222	10857
Undersampling	290	0.0	42110	13957
Oversampling	295	0.002	5568459	1856828

All 4 models fail to find any meaningful information in our data, which can be seen in the poor R2 metrics we obtained.

So how is this possible? We need to do a bit more exploration in our data to find possible answers.

One important characteristic for predicting data is the correlation between our features and our label, if the features do not correlate with the data, then we won't be able to recognize any patterns.

To see this graphically, we calculated a correlation matrix, using only the continuous variables (numerical) and the claim amount (label).

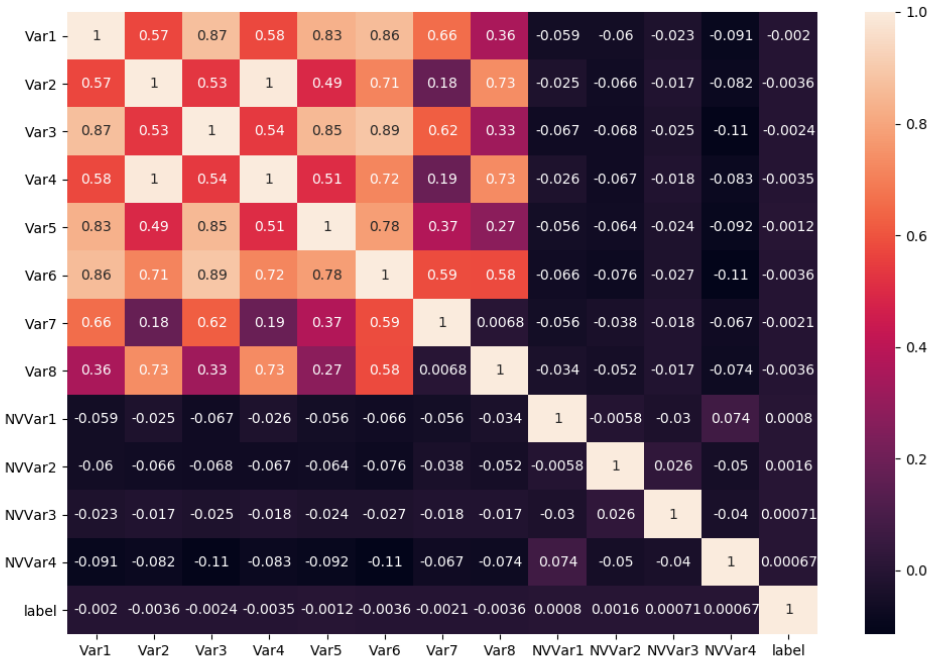
In this correlation matrix we can see that there is correlation between some of our variables (Var1-Var8). Also we can see in the last row the correlation between our value to predict and all the variables, which is very close to 0.

This shows us that **none** of our features correlate with our label, which is why our models have no capacity to predict the values. One of the possible explanations for this is that these car characteristics are simply not a very good indicator for the amount of a claim, which could be much more correlated with some characteristics of the driver of the car (age for example).

Logistic + Gamma Regression

For this second part, we try to model this data in a two step process, first we will fit a classification model that can tell us if there is going to be a claim, and then if there is a claim, fit a gamma regression to estimate the value.

We tried our different datasets with a random forest classifier, where it was clear that the Oversampled dataset was the getting a better Accuracy and AOC. In the end the best model we could train was a random forest with 20 trees and a maxdepth of 30, this model managed to score an 84% accuracy on the oversampled dataset, but much of this accuracy might be due to overfitting.



One important aspect of gamma regression is that the response variable (Y) must be greater than 0, so at first we were getting some errors when the logistic regression predicted a claim with a 1, but the real value was a 0.

This is why we had to filter only the predicted results with a 1, that had a concurrent label as 1 (true positive).

Then we fitted a gamma regression using our pcaFeatures and the claim values, other parameters used were inverse link and 10 iterations.

The r2 result for this prediction was a **-0.20** R-squared which is very bad and has worse predictability than just predicting the sample mean.

The total running time with 10 cores was around 17 minutes, and with 20 cores it improved to 11 minutes, this is probably because this task is much more parallelizable than the first question.

After this extensive analysis I can only conclude that this dataset does not contain features that are representative of the amount value of an insurance claim, and if we want to model this with more predictive features, we could take a look at the characteristics of the drivers of the cars that incur in insurance claims.