Q 1.1 I choose to plot the "*Average Blood Glucose Level per Age Group (see Figure 1)"*, which shows how blood glucose level, on average, increases with age. The *"Average Hemoglobin Level per Age Group (see Figure 2)*" is also plotted, and it shows how the hemoglobin level, on average, decreases with age.

I have chosen these graphs to depict how the long-term exposure to tech is literally sucking the life out of us. Increases in blood glucose level and decrease in the hemoglobin level is not good at all. The target demographic will be alarmed when they see these results.

Q 2.1 To begin with, I imputed missing values for all features who had missing values in them. I dropped all columns with IDs and columns with zero signal such as gender and country. After imputation and dropping some variables I had these features which I used: 'age', 'zodiac_sign', 'web_browser', 'state', 'smoker', 'drinker', 'weight_in_kg', 'waistline_in_cm', 'is_obese', 'is_severely_obese', 'bmi', 'Blood Glucose Level', 'Blood Serum Creatinine', 'Cholesterol (Total)', 'Diastolic Blood Pressure', 'Gamma Glutamyl Transferase', 'Hemoglobin', 'High Density Lipoprotein Cholesterol', 'Low Density Lipoprotein Cholesterol', 'Proteinuria', 'Serum Glutamic Oxaloacetic Transaminase (ALT)', 'Serum Glutamic Oxaloacetic Transaminase (AST)', 'Systolic Blood Pressure', 'Triglycerides Level', 'left_eye_sight_correction', 'right_eye_sight_correction', 'left_ear_hearing_status', 'right_ear_hearing_status', 'elemental_group', 'height_in_centimeters'. Basically almost all available features. To prevent overfitting I considered dropping some variables that were highly correlated such as 'bmi' and 'waistline_in_cm', however after running the initial model of XGBoost and Logistic Regression with all of these features it was evident that overfitting was not a problem since the difference between the train accuracy and test accuracy was almost zero. Thus, I decided to keep them.

Moving on the model. I tried a default (i.e., without any hyperparameters) KNN, Logistic Regression, Single Decision Tree, Random Forest, XGBoost and Support Vector Machine. From the first initial run, I discarded all but the XGBoost and Logistic Regression since they had among the highest accuracies and the by far the lowest difference between the train accuracy and test accuracy. For the *"Output XGBOOST with default parameters (see Table 1)"* the test accuracy was higher than the test accuracy from the *"Output Logistic Regression with default parameters (see Table 2)"*. Then I hyperparametered tuned both models. Comparing the *"Output Logistic Regression after hyperparameter tuning (see Table 4)"* with the default Logistic Regression is is evident that the hyperparameterization did not improve the model at all. However, the XGBoost Model became slightly better and these results are shown in *"Output XGBOOST after hyperparameter tuning (see Table 3)"*. Based on this, my final model became the XGBOOST with the parameters to be tuned where 'subsample', 'n_estimators', 'max_depth',' learning_rate' and 'colsample_bytree'. To tune these parameters I used a RandomizedSearchCV process. In *"Parameter grid for XGBOOST hyperparameter tuning (see Table 5*)" the exact values for each parameter are presented. My main two parameters to be tuned were the 'n_estimators' and the 'max_depth'. 'n_estimators' refers to the number of boosting stages to perform. This is important since choosing too many trees (would likely lead to overfitting) and too few trees (might lead to underfitting). Similarly, 'max_depth', which defines the maximum number of nodes from the root to the farthest leaf, was important since if it is too shallow we run the risk of underfitting and too deep then we run the risk of overfitting. Lastly, learning_rate, the subsample and colsample_bytree were chosen based on the model from the Housing Market Predictor Assignment, with no real thought behind why.

Q 2.2 *In the "Confusion Matrix to Evaluate Drinker Model Performance (see Figure 3)",* the hyperparameter tuned XGBoost model performs rather well. The share of correctly predicted Drinkers and Non-drinkers are higher than the mispredicted. Also, the true classes of outcomes are evenly distributed in the confusion matrix, where the correct prediction for the Non-Drinkers is 2.7 times higher than wrongly predicted Non-drinkers, and 2.5 for the correct Drinker prediction against the wrongly predicted Drinkers. This reflects the fact that our target variable 'drinker' was balanced in the training and test set.

Q 2.3 I show the five most important features in the "*Top Ten Features Importance by Weight (see Figure 4)"*. By weight I mean the number of times a feature is used to split the data across all trees. I included five extra features, to show how relatively more frequent the top five were against other less frequent features. The top five features were: 'Gamma Glutamyl Transferase', 'High Density Lipoprotein Cholesterol', 'Serum Glutamic Oxaloacetic Transaminase (ALT)', 'Triglycerides Level' and 'heigth_in_centimeters'. An educated guess on why they are so predictive is the alcohol would either systematically increase (or decrease) the blood test values. This effect is consequently picked up by the model. For 'heigth_in_centimeters' the assumption that short people drink more/less than taller people is not reasonable. However, It would be reasonable if a person's social background is reflected through 'heigth_in_centimeters'. Healthier kids often turn out taller than unhealthy kids, and this is often associated with the resources the kids' families are. A hypothesis based on this would specify that taller people often come from higher socio-economic standards and thus when adults have more resources to spend on alcohol.

Q 3.4

Firstly, if the numbers are to be trusted or not relies on drinker and smoker classification. For the drinker classification I am confident that our model will be able to somewhat accurately predict drinkers and non-drinkers. However, for the smoker classification, the results are not as straightforward since the target variable was highly unbalanced. I had to artificially oversample the minority to be able to predict smokers. My model is better at predicting the correct class of smokers than just a random classification, but it is not by much. This is evident if looking at the "*Confusion Matrix to Evaluate Smoker Model Performance (see Figure 5)".* This highly impacts the probability of incorrect classification, which is detrimental to the selling analysis.

For the selling analysis, I calculated the expected values using the values from the confusion matrix to estimate the probabilities and took into account the value counts for each persona. Given this, I tried to decide whether or not to sell a specific person to each company.

Firstly, for Johnny Talker, only Sinners and Tipplers had a positive expected profit and by selling them I expect to make 29830 monetary units. Secondly, for Rat Race Ltd, it was more complex since the calculation would have to involve the probabilities of the drinker and the smoker classifications simultaneously. I only (hopefully) managed to calculate it for the saints, and if I only sell saints to Rat Race Ltd, then I expect to make 42630 monetary units. Finally, for Spin2Win I wanted to calculate the probability of having at least one vice, but I did not manage to do it. Nonetheless, I decided to include Sinners Fumers and Tipplers. Since both our models are better predictors than a random 50/50 predictor, then I would expect to make 874 monetary units.

Appendix

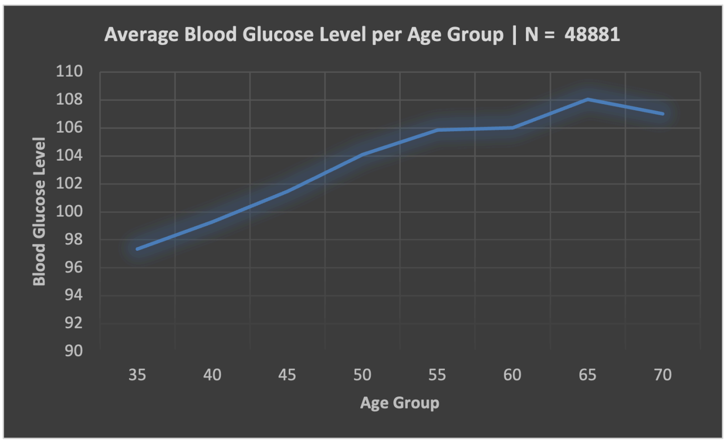


Figure 1: Average Blood Glucose Level per Age Group

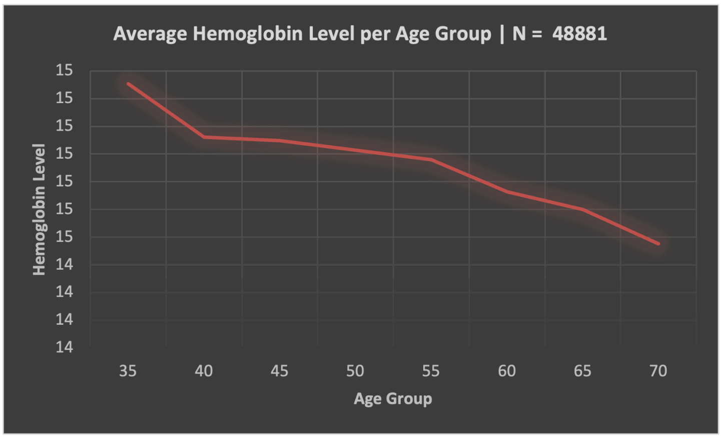
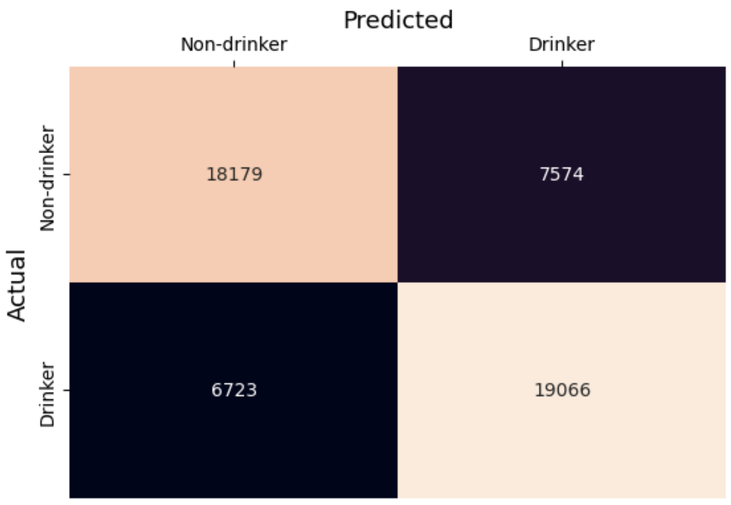


Figure 2: Average Hemoglobin Level per Age Group



	Non-drinker	Drinker
Non-drinker	18179	7574
Drinker	6723	19066

Figure 3: Confusion Matrix to Evaluate Drinker Model Performance

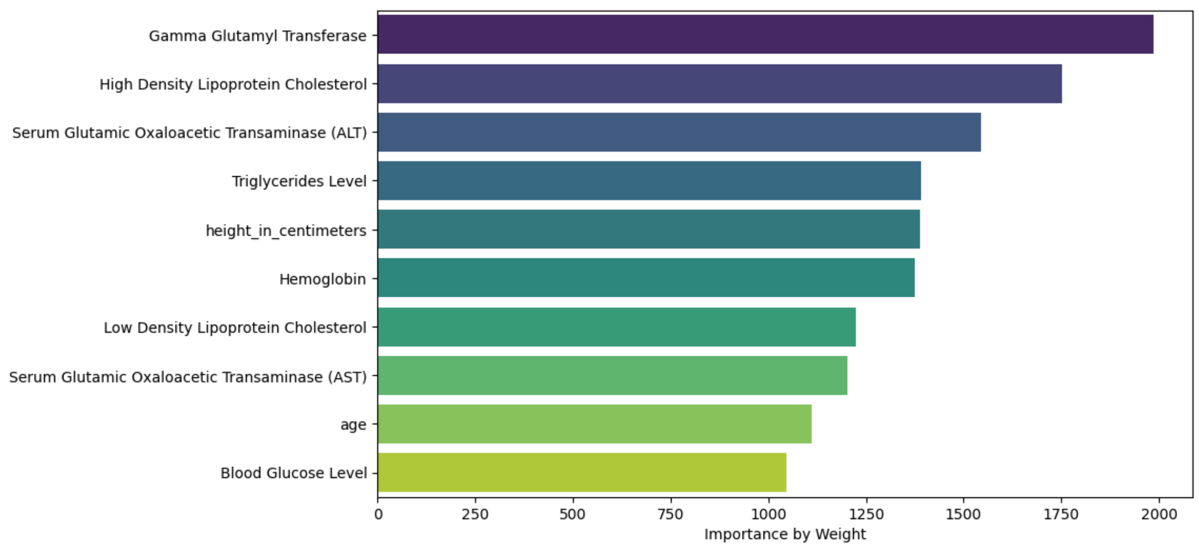


Figure 4: Top Ten Features Importance by Weight

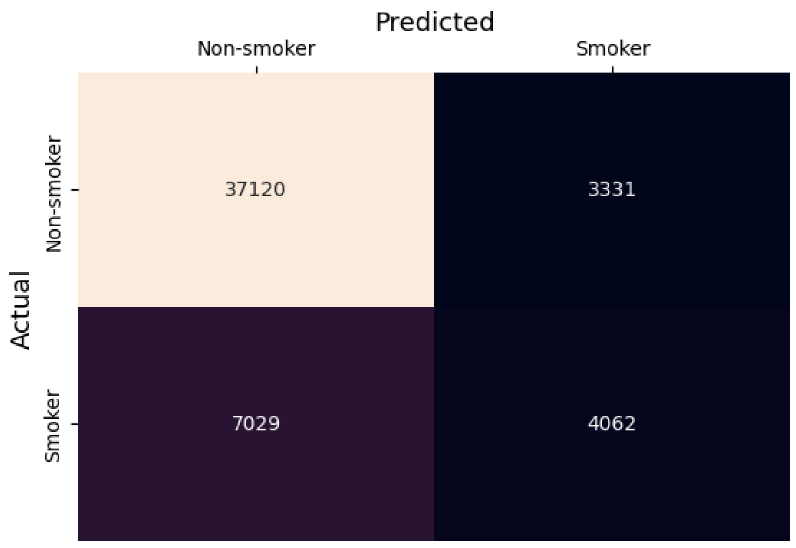


Figure 5: Confusion Matrix to Evaluate Smoker Model Performance

```
Default Parameters: {'subsample': 1.0, 'n_estimators': 100, 'max_depth': 6,
'learning_rate': 0.03, 'colsample_bytree': 1.0}
Training: 0.7348
Testing: 0.7213
```

Table 1: Output XGBOOST with default parameters

```
Default Parameters: {'solver': 'lbfgs', 'penalty': 'l2', 'max_iter': 100,
'class_weight': None, 'C': 1.0}
Training: 0.7039
Testing: 0.7040
```

Table 2: Output Logistic Regression with default parameters

```
Best Parameters: {'subsample': 1.0, 'n_estimators': 400, 'max_depth': 6,
'learning_rate': 0.05, 'colsample_bytree': 0.6}
Training: 0.7302
Testing: 0.7226
```

Table 3: Output XGBOOST after hyperparameter tuning

```
Best Parameters: {'solver': 'saga', 'penalty': 'l1', 'max_iter': 100,
'class_weight': 'balanced', 'C': 0.01}
Training: 0.7039
Testing: 0.7038
```

Table 4: Output Logistic Regression after hyperparameter tuning

```
param_grid = {
   'max_depth': [4, 5, 6, 8],
   'subsample': [0.6, 0.8, 1.0],
   'colsample_bytree': [0.6, 0.8, 1.0],
   'learning_rate': [0.01, 0.05, 0.1],
   'n_estimators': [100, 200, 400, 600]}
```

Table 5: Parameter grid for XGBOOST hyperparameter tuning