CS 6923 Machine Learning
Spring 2019
Final Project Report

Name: Kaustuk Kumar
NetID: kk3697

Name: Nikhil Narasimha Prasad
NetID: nnp267

PART I: Preprocessing (No more than two pages for this part)

1. How does your program handle missing values? And why?

- **The features that have missing values are "house", "weight", "player_code", "move_specialty".**
- **Missing data can be treacherous because it is difficult to identify the problem. It is not always obvious when missing data will cause a problem.**
- **Missing data can also lead to misleading results by introducing bias. Whenever segments of the target population do not respond, they become under-represented in the data. In this situation, we end up not analyzing what we intended to measure.**
- **Weight is missing in over approximately 97% records. These missing values are poorly interpretable and cannot be generalized to other players. Therefore, this feature is just dropped completely.**
- **For the other features, we use an "Imputer" function which replaces all the missing values with the median of all the values in the feature.**

```
from sklearn.preprocessing import Imputer

imp = Imputer(missing_values = 'NaN', strategy = 'median', axis = 0)
imp.fit(X_test_new)
X_test_new= pd.DataFrame(data = imp.transform(X_test_new), columns = X_test_new.columns)
```

2. If your program converts numeric features to categorical features, or categorical features to numeric features, describe how it does it.

- **There are multiple categorical features in the data, which need to be converted to numeric features to run our classification models.**
- **The categorical features that need to be converted are "house", "gender", "player_code", "move_specialty", "player_type", "snitchnip", "stooging", "change" and "snitch_caught".**
- **We use cat.codes to assign numeric codes to these categorical features as show below.**

```
X_new['house'] = X_new['house'].astype('category')
X_new['gender'] = X_new['gender'].astype('category')
X_new['player_code'] = X_new['player_code'].astype('category')
X_new['move_specialty'] = X_new['move_specialty'].astype('category')
X_new['player_type'] = X_new['player_type'].astype('category')
X_new['snitchnip'] = X_new['snitchnip'].astype('category')
X_new['stooging'] = X_new['stooging'].astype('category')
X_new['change'] = X_new['change'].astype('category')
X_new['snitch_caught'] = X_new['snitch_caught'].astype('category')
```

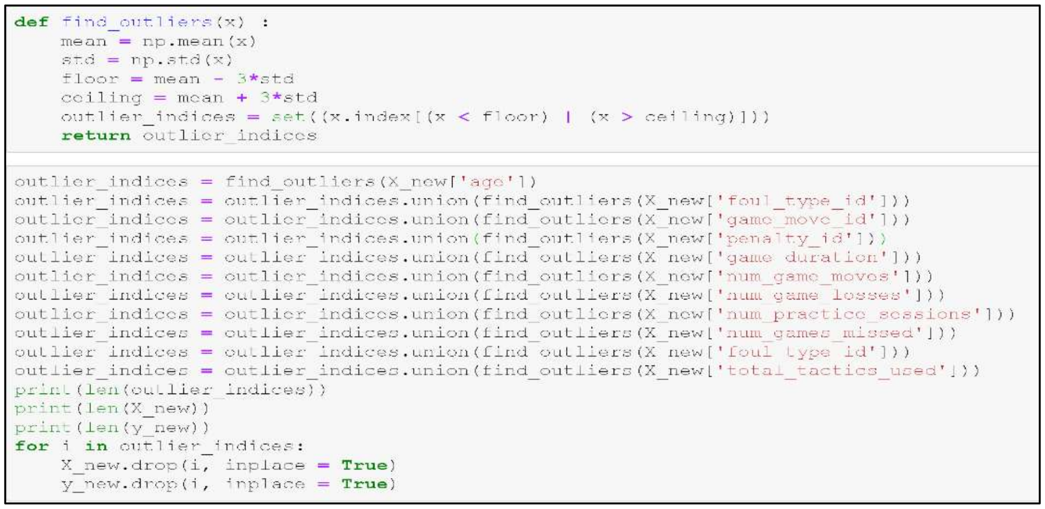
```
cat_columns = X_new.select_dtypes(['category']).columns
X_new[cat_columns] = X_new[cat_columns].apply(lambda x: x.cat.codes)
X_new.info()
```

3. Describe any feature selection, combination or creation, and any feature values combination performed by your program and the reasons for doing so.

- **The features “num_games_satout”, “num_games_injured” and “num_games_notpartof” can be combined to create a new feature which indicates that the player did not participate.**
- **We create a new feature called “num_games_missed”, which is the sum of the above three features. After creating the new feature, we can drop the previous 3 features, hence simplifying our model.**
- **We calculate the total number of tactics used by every player and create a new feature called “total_tactics_used” from the 23 features for every tactic used by a player.**
- **We take the count of all the tactics features that have a value which is not equal to “No” and store the count in the new feature “total_tactics_used”.**
- **The feature “total_tactics_used” is then used for any future preprocessing or model fitting, and the 23 features for tactics are dropped from the dataset. This also helps in simplifying the model even further.**

4. Describe other preprocessing used in your program (e.g. centralizing, normalization)

- **We reduce the number of categories for the features “stooging” and “snitchnip”. These features are recategorized into 3 categories “normal”, “none” and “high”.**
- **We perform Log Transformation for the feature that is the most skewed in the dataset, i.e., “num_games_missed”.**
- **Since we used log transformation, we standardize the “num_games_missed” feature using the formula:
  New value = (Value - Mean(values)) / StandardDeviation(values)**
- **We remove the outliers for all the numerical features which helps us in getting a more relevant mean and standard deviation.**
- **We restrict the data to 3 standard deviations on either side from the mean for each numeric value, as shown below.**
- **Since the data is very imbalanced, we use SMOTE to make the data balanced.**



```python
def find_outliers(x) :
    mean = np.mean(x)
    std = np.std(x)
    floor = mean - 3*std
    ceiling = mean + 3*std
    outlier_indices = set((x.index[(x < floor) | (x > ceiling)]))
    return outlier_indices
```

```python
outlier_indices = find_outliers(X_new['age'])
outlier_indices = outlier_indices.union(find_outliers(X_new['foul_type_id']))
outlier_indices = outlier_indices.union(find_outliers(X_new['game_move_id']))
outlier_indices = outlier_indices.union(find_outliers(X_new['penalty_id']))
outlier_indices = outlier_indices.union(find_outliers(X_new['game_duration']))
outlier_indices = outlier_indices.union(find_outliers(X_new['num_game_moves']))
outlier_indices = outlier_indices.union(find_outliers(X_new['num_game_losses']))
outlier_indices = outlier_indices.union(find_outliers(X_new['num_practice_sessions']))
outlier_indices = outlier_indices.union(find_outliers(X_new['num_games_missed']))
outlier_indices = outlier_indices.union(find_outliers(X_new['foul_type_id']))
outlier_indices = outlier_indices.union(find_outliers(X_new['total_tactics_used']))
print(len(outlier_indices))
print(len(X_new))
print(len(y_new))
for i in outlier_indices:
    X_new.drop(i, inplace = True)
    y_new.drop(i, inplace = True)
```

PART II: Classification (No more than two pages for each model in this part)

Model One:

1. **Supervised learning method used in this model is**
   <u>Logistic Regression</u>

2. **Why you choose this supervised learning method?**
   - **Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.**
   - **We choose this model to estimate a baseline accuracy for the preprocessed dataset.**
   - **We assess the outputs of the data to make a decision if a more complex model is required.**

3. **Describe the method you used to evaluate this method.**
   - **Accuracy:**
     - **Accuracy = (TP + TN) / (TP + TN + FP + FN)**
     - **We use the accuracy_score() method provided by sk-learn to get the accuracy of the model on the the given data.**
   - **Confusion matrix**
     - **A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.**
     - **We use the confusion_matrix() method provided by sk-learn to get the confusion matrix of the model on the given data.**

4. **Describe process of experimenting different parameter settings or associated techniques.**
   - **Parameter name: class_weight (Weights associated with classes in the form {class_label: weight})**
     - **Parameter values: None, balanced**
     - **Performance of different values:**
       - **Class_weight = None, Accuracy = 48.28%**
       - **Class_weight = balanced, Accuracy = 48.28%**
     - **Analysis:**
       - **The performance is similar for both the values in the case of logistic regression.**

- **Parameter name: C (Inverse of regularization strength)**
    - **Parameter values: 0.01, 1, 100**
    - **Performance of different values:**
        - **C = 0.01, Accuracy = 48.28%**
        - **C = 1, Accuracy = 48.28%**
        - **C = 100, Accuracy = 48.28%**
    - **Analysis:**
        - **The performance is similar for both the values in the case of logistic regression.**
- **Parameter name: folds (Number of folds in K-Fold Cross Validation)**
    - **Parameter values: 5, 10, 15, 20**
    - **Performance of different values:**
        - **folds = 5, Accuracy = 45.43%**
        - **folds = 10, Accuracy = 46.18%**
        - **folds = 15, Accuracy = 47.46%**
        - **folds = 20, Accuracy = 48.28%**
    - **Analysis:**
        - **The performance of this model improves as the number of folds increases.**

5. Accuracy and Confusion matrix with most suitable parameters

		Predicted	
		-	+
Actual	-	**44143**	**38205**
	+	**46973**	**35375**

Accuracy: 48.28%

Model Two:

1. **Supervised learning method used in this model is <u>Decision Tree Classifier</u>**

2. **Why you choose this supervised learning method?**
    - **The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from training data.**
    - **Decision trees implicitly perform variable screening or feature selection.**
    - **Decision trees require relatively little effort from users for data preparation.**
    - **Nonlinear relationships between parameters do not affect tree performance.**

3. **Describe the method you used to evaluate this method.**
    - **Accuracy**
        - **Accuracy = (TP + TN) / (TP + TN + FP + FN)**
        - **We use the accuracy_score() method provided by sk-learn to get the accuracy of the model on the the given data.**
    - **Confusion matrix**
        - **A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.**
        - **We use the confusion_matrix() method provided by sk-learn to get the confusion matrix of the model on the given data.**

4. **Describe process of experimenting different parameter settings or associated techniques.**
    - **Parameter name: leaves (Number of leaves in the decision tree)**
        - **Parameter values: 10, 30, 100, 500**
        - **Performance of different values:**
            - **leaves = 10, Accuracy = 91.43%**
            - **leaves = 30, Accuracy = 91.82%**
            - **leaves = 100, Accuracy = 91.37%**
            - **leaves = 500, Accuracy = 88.98%**

- **Analysis:**
    - **Initially the performance of the model improves with increase in number of leaves, peaks at 30, and then the accuracy decreases with increase in number of leaves.**

- **Parameter name: min_samples_split (Minimum number of samples required to split an internal node)**
    - **Parameter values: 2, 5, 10**
    - **Performance of different values:**
        - **min_samples_split = 2, Accuracy = 91.82%**
        - **min_samples_split = 5, Accuracy = 91.82%**
        - **min_samples_split = 10, Accuracy = 91.82 %**
    - **Analysis:**
        - **The performance is similar for all values of min_samples_split**
- **Parameter name: max_depth (The maximum depth of the tree)**
    - **Parameter values: 10, 30, 100**
    - **Performance of different values:**
        - **max_depth = 10, Accuracy = 89.58%**
        - **max_depth= 30 Accuracy = 91.82%**
        - **max_depth = 100, Accuracy = 91.82 %**
    - **Analysis:**
        - **Initially the accuracy increases with increase in depth of the tree, but stagnates after a depth of 30.**

5. Accuracy and Confusion matrix with most suitable parameters

		Predicted	
		-	**+**
Actual	**-**	**16084**	**302**
	+	**2391**	**14163**

Accuracy: 91.82%

Model Three:

1. **Supervised learning method used in this model is**
   <u>Random Forest Classifier</u>

2. **Why you choose this supervised learning method?**
   - **Random Forest builds multiple trees and combines them together to get a more accurate result.**
   - **Random forest classifier will handle the missing values and maintain the accuracy of a large proportion of data.**
   - **If there are more trees, it won't allow overfitting trees in the model.**
   - **It has the power to handle a large data set with higher dimensionality**

3. **Describe the method you used to evaluate this method.**
   - **Accuracy**
     - **Accuracy = (TP + TN) / (TP + TN + FP + FN)**
     - **We use the accuracy_score() method provided by sk-learn to get the accuracy of the model on the the given data.**
   - **Confusion matrix**
     - **A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.**
     - **We use the confusion_matrix() method provided by sk-learn to get the confusion matrix of the model on the given data.**

4. **Describe process of experimenting different parameter settings or associated techniques.**
   - **Parameter name: n_estimators (The number of trees in the forest)**
     - **Parameter values: 100, 200, 300**
     - **Performance of different values:**
       - **n_estimators = 100, Accuracy = 94.00%**
       - **n_estimators = 200, Accuracy = 93.95%**
       - **n_estimators = 300, Accuracy = 94.05%**
     - **Analysis:**
       - **There is a minimal increase in the accuracy as the number of trees increases.**

- **Parameter name: max_depth (The maximum depth of the tree)**
    - **Parameter values: 30, 70, 90**
    - **Performance of different values:**
        - **max_depth = 30, Accuracy = 94.00%**
        - **max_depth= 70 Accuracy = 94.05%**
        - **max_depth = 90, Accuracy = 94.05%**
    - **Analysis:**
        - **There is a minimal increase in the accuracy as the max_depth increases and then in stagnates after 70.**
- **Parameter name: min_samples_leaf (The minimum number of samples required to be at a leaf node)**
    - **Parameter values: 1, 2, 4**
    - **Performance of different values:**
        - **min_samples_leaf = 1, Accuracy = 94.05%**
        - **min_samples_leaf = 2 Accuracy = 94.00%**
        - **min_samples_leaf = 4, Accuracy = 93.95%**
    - **Analysis:**
        - **The accuracy of the model decreases as the number of min_samples_leaf increases.**

5. Accuracy and Confusion matrix with most suitable parameters

		Predicted	
		-	+
Actual	**-**	**16346**	**40**
	+	**1919**	**14635**

Accuracy: 94.05%

PART III: Best Hypothesis (No more than two pages for each model in this part)

1. **Which model do you choose as a final method?**

   Model number: <u>Three</u>

   Supervised learning method used in this model: <u>Random Forest Classifier</u>

2. **Reasons for choosing this model.**
   - **This model gives us the best score for accuracy on the training data.**
   - **This model also gives us the best confusion matrix amongst all the other models. It has the least number of False Positives and False Negatives.**

3. **What are the reasons do you think that make it has the best performance?**
   - **Random forests are a strong modeling technique and much more robust than a single decision tree or logistic regression.**
   - **Random forests aggregate many decision trees to limit overfitting as well as error due to bias and therefore yield better results.**