Vivian-Maiyo /
dsc-phase-3-project

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

MIT license

0 stars 0 forks 1 watching 1 Branch 0 Tags Activity

Public repository

main 1 Branch 0 Tags Go to file t Go to file Add file Code

Vivian-Maiyo Finished Project 7 minutes ago

.gitignore	Initial commit	23 minutes ago
CONTRIBUTING.md	Initial Commit	21 minutes ago
Group 12 Phase3 presentation .p...	Finished Project	7 minutes ago
Group 12 Phase3 presentation .p...	Finished Project	7 minutes ago
Group12Project Final.ipynb	Finished Project	7 minutes ago
LICENSE	Initial commit	23 minutes ago
README.md	Finished Project	7 minutes ago
notebook.pdf	Finished Project	7 minutes ago

README MIT license

TANZANIA WATER WELLS ANALYSIS

OVERVIEW

The Tanzanian government, foreign donor organizations, churches and faith-based organizations, and even the villagers have all been involved in funding and installation of wells. Some of the installed wells however cease to function over time while others remain in dilapidated conditions needing repair.

For this project i'll be working as a Data scientist along an NGO alligned towards provision of water among communities in Tanzania. I'm to analyse existing data within the Tanzanian water ministry and create a model that will guide us on the functionality estimated life span of wells in different regions and varying conditions.

TOOLS

Python Jupyter Notebook

Python Libraries used

```
Pandas: Data pre_processing and cleaning
Matplotlib: Data visualization
Seaborn: Data visualization
Numpy: Data pre_processing
```

ML ALGORITHM used

```
Scikit_learn: Modelling
```

WORKFLOW

1. Data preparation

- Imported the necessary packages
- Loading the training-set-values dataset as downloaded
- Checked for the general data shape
- Concatenating the training and testing data sets.
- Checking for duplicates in the dataset.
- Checking and handling missing values in our dataset.
- Feature engineering.
- Dropping columns that are of low impact to model outcome.

2. Data analysis & visualization

Conducted concurrent analysis and visualization, i'm trying to show the relationship between different columns in our dataset.E.g relationship between the ['funder'& 'region']. Displayed the relationship between the numerical data E.g ['amount_tsh', 'gps_height', 'population', 'age_of_well']

https://github.com/Lewis-Gitari/Tanzania-water-wells

3. Modelling

We'll first identify the categorical and numerical columns for ease of one hot encoding to allow easy fitting to different models.

- categorical data _columns = ['funder','basin','region', 'public_meeting', 'permit', 'extraction_type_group','management','payment_type','water_quality', 'quantity','source','waterpoint_type_group']
- numerical data_columns = ['amount_tsh', 'gps_height', 'population', 'age_of_well']

After encoding our data results into (56991, 106) shape, 56991 rows, 106 columns. I then split the data before we carry out any further transformations to prevent data leakage. For the purposes of splitting, we'll use the test_size 0.25 and random_state 42

a) We'll start by fitting a dummy classifier somewhat equal to guessing as a baseline model that we can how the rest of our models perform with.

```
Our dummy model has an accuracy score of 0.448 which means it makes correct predictions 44% of the
times. This is worse than random guessing.
```

b) Simple Logistic Regression Model

```
Our simple logistic regression model has an accuracy of 0.738. This means that it is predicting
correctly 73% of the time. This is not bad for a vanilla model without any tuning.
```

c) We'll tune the Logistic regression further by adding an intercept and setting a high regularization parameter to see how it will perform

```
Logistic Tuned Training Accuracy:  0.7364714690124698
Logistic Tuned Validation Accuracy:  0.7344890510948905
Even with the tuning, our accuracy has remained constant and although the model has registered
slight improvement in its ability to classify functional wells that need repair, it is still not
very good.
```

d) DecisionTreeClassifier

```
CLF Baseline Training Accuracy:  0.9482020447792621
CLF Baseline Validation Accuracy:  0.7489472206625492
This model has a training accuracy of 0.81 and a validation accuracy of 0.74.
```

e) We are going to make use of GridSearchCV to find an optimal combination of parameters that we can use to tune the model.

```
CLF Tuned Training Accuracy:  0.8159698664108743
CLF Tuned Validation Accuracy:  0.7606681639528355
Tuning our model has slightly improved it. While training accuracy remains 0.81, validation accuracy
has come up to 0.76. The closeness of the two values suggests the model is not overfitting.
```

f) KNeighborsClassifier

```
KNN Training Accuracy:  0.812834850150902
KNN Validation Accuracy:  0.7501403705783268
One of the ways of tuning a KNN model is by iterating through various values of K to find the value
that gives the best results. We will make use of grid search to iterate through several values of k
to find an optimal value of k that we can use to tune the model.
```

g) Bagged Tree and Random Forest Models

```
Forest Training Accuracy:  0.68689609994619
Forest Validation Accuracy:  0.6843767546322291
While bagged tress and random forests have not registered the highest accuracy scores, they are so
far the ones with the most consistent performance in both the training and test sets.

We will use GridSearchCV to find optimal values to tune the parameters.
Refitting the model with the best parameters identified by GridSearchCV
    tuned_forest = RandomForestClassifier(bootstrap=True, criterion='entropy', max_depth=20,
min_impurity_split=0.1)
    tuned_forest.fit(X_train_scaled, y_train)
Tuned Forest Training Accuracy:  0.8968018155019535
Tuned Forest Validation Accuracy:  0.7911285794497473
```

FINAL MODEL AND CONCLUSION.

Final Model Of all the models tested, the tuned random forest model has given the best results at a training accuracy of 0.89 and a test accuracy of 0.79. We'll tune it a little further in this section and then present it as our final predictive model.

The most important features in predicting well function as depicted by our final model are

- Altitude of the well (gps_height)
- Well age (age_of_well)
- Population (population)
- Enough water quantity (quantity_enough)
- Amount of water in total static head (amount_tsh)
- Insufficient water quantity (quantity_insufficient)
- The waterpoint kind (waterpoint_type_group_other)
- The kind of extraction used by the waterpoint(extraction_type_group_other)
- Well having a permit(permit_True)

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

- **Jupyter Notebook** 100.0%