cmps320_ml_nischalbhandari_assignment3.ipynb

October 31, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

import sklearn.linear_model as skl_lm
from sklearn.metrics import confusion_matrix, classification_report,␣
 ↪precision_score
from sklearn import preprocessing


import statsmodels.api as sm
import statsmodels.formula.api as smf

%matplotlib inline
plt.style.use('seaborn-white')
```

[2]:
```
success_df = pd.read_csv("ex12-40.txt", index_col= None)
success_df.head()
```

[2]:
```
   'Experience'  'Success'
0             2          0
1             4          0
2             5          0
3             6          0
4             7          0
```

[3]:
```
success_df.rename(columns={"'Experience'": 'Months of Experience', "'Success'" :
 ↪ "Success" }, inplace=True)
success_df.head()
```

[3]:
```
   Months of Experience  Success
0                     2        0
1                     4        0
2                     5        0
3                     6        0
```

```
4                     7         0
```

[5]:
```
success_df.shape
```

[5]:
```
(25, 2)
```

0.0.1 Determine whether months of experience is associated with the probability of completing the task.

[36]:
```
corr_exp_success = success_df.corr(method='pearson')
# not showing in jupyter's pdf
corr_exp_success = corr_exp_success.style.background_gradient(cmap='coolwarm')
corr_exp_success
```

[36]:
```
<pandas.io.formats.style.Styler at 0x144516b50>
```

	Months of Experience	Success
Months of Experience	1.000000	0.439664
Success	0.439664	1.000000

There is a positive linear relationship between “Months of Experience” and “Success” in the dataset as indicated by the pearson correlation coefficient of 0.440.

[7]:
```
#  check for p-value for the above-mentioned correlation coeff
import scipy.stats as stats

p_value = stats.pearsonr(success_df['Months of Experience'],␣
 ↪success_df['Success'])[1]
print('p-value:', p_value)
```

```
p-value: 0.027866370536279778
```

If we choose the p-value threshold of 0.05, the calculated p-value is less than the threshold. So we can say there is significant correlation between the month of experience and the probability of completing the task.

0.0.2 Compute the probability of completing the task for an engineer having 36 months of experience

[9]:
```
# building a regression model

from sklearn.linear_model import LogisticRegression
```

```
X = success_df['Months of Experience'].values.reshape(-1, 1)
y = success_df['Success']

logistic_reg = LogisticRegression()
logistic_reg.fit(X, y)
```

[9]:
```
LogisticRegression()
```

[10]:
```
print('classes: ',logistic_reg.classes_)
print('intercept :', logistic_reg.intercept_)
print('coefficient: ',logistic_reg.coef_)
```

```
classes:  [0 1]
intercept : [-1.6783687]
coefficient:  [[0.11896832]]
```

We have positive regression coefficient which means an increase in the variable is associated with an increase in the log-odds of the event occurring. So one month increase in experience of an engineer is associated with an increase in the log odds of successful task completion by 0.119 units.

Using the previous model to predict the probability for the completion of a task

[17]:
```
X_new = np.array([36]).reshape(-1,1)
```

[18]:
```
y_pred = logistic_reg.predict(X_new) # default threshold is 0.5
y_pred
```

[18]:
```
array([1])
```

The prediction is 1 which means that an engineer with 36 months of experience will complete the given task. ***Now let's calculate the probabilit***

[19]:
```
probabilities = logistic_reg.predict_proba(X_new)[:,1] # request a response␣
  ↪from the logistic regression estimator with probability
probabilities
```

[19]:
```
array([0.93115005])
```

The probability of completing the task for an engineer having 36 months of experience is **93 %**.

0.1 Exam Problem 7.

Steps to Develop a House Price Prediction Model:

1. Data Collection and Preprocessing:

Explore the data for missing values, outliers, and inconsistencies. Clean the data by handling missing values, normalizing numerical features, encoding categorical variables, and addressing outliers.

2. Feature Selection and Engineering:

Select relevant features based on their impact on house sale price. Use the best subset selection and regularization method like elastic net.

Perform feature engineering if needed, such as creating new features (e.g., price per square foot, age of the property).

3. Split the Data:

Split the dataset into training and testing sets to evaluate the model's performance. I will split data into 80 % training and 20 % testing sets.

4. Choose a Model: Initially, I will default to linear regression as I assume we have lots of numerical data in the dataset.
5. Model Training: Train the selected model on the training data using the chosen features.

Tune hyperparameters using techniques like cross-validation or grid search to optimize the model's performance.

6. Model Evaluation: Evaluate the model's performance on the test set using appropriate metrics (e.g., Mean Absolute Error, Root Mean Squared Error, R-squared).

Analyze the residuals to ensure the model's predictions do not exhibit systematic errors.

7. Fine-tuning and Validation

Make adjustments to the model, such as trying different models or feature combinations to improve performance.

Validate the model's performance using cross-validation techniques to ensure it generalizes well to new data.

8. Deployment and Monitoring:

Once I find the model performing satisfactorily, I will deploy it into the real estate firm's system.

9. Performance Metric Analysis:

I would use the following two metrics to evaluate the model's performance:

Root Mean Squared Error (RMSE): It provides a measure of how well the model fits the actual values, giving higher weight to larger errors.

R-squared (R^2): It indicates the proportion of variance in the house prices that is predictable by the model.