Machine Learning Worksheet – 6 Answers

1) B
2) C
3) D
4) B
5) B
6) A & D
7) B & C
8) A & C
9) B
10) Explain how does the adjusted R-squared penalize the presence of unnecessary predictors in the model?

The predicted R-squared indicates how well a regression model predicts responses for new observations. This statistic helps you determine when the model fits the original data but is less capable of providing valid predictions for new observations. (Read an example of using regression to make predictions.)

Minitab calculates predicted R-squared by systematically removing each observation from the data set, estimating the regression equation, and determining how well the model predicts the removed observation. Like adjusted R-squared, predicted R-squared can be negative and it is always lower than R-squared.

Even if you don’t plan to use the model for predictions, the predicted R-squared still provides crucial information.

A key benefit of predicted R-squared is that it can prevent you from overfitting a model. As mentioned earlier, an overfit model contains too many predictors and it starts to model the random noise.

Because it is impossible to predict random noise, the predicted R-squared must drop for an overfit model. If you see a predicted R-squared that is much lower than the regular R-squared, you almost certainly have too many terms in the model.

You can try these examples for yourself using this Minitab project file that contains two worksheets. If you want to play along and you don't already have it, please download the free 30-day trial of Minitab Statistical Software!

There’s an easy way for you to see an overfit model in action. If you analyze a linear regression model that has one predictor for each degree of freedom, you’ll always get an R-squared of 100%!

In the random data worksheet, I created 10 rows of random data for a response variable and nine predictors. Because there are nine predictors and nine degrees of freedom, we get an R-squared of 100%.

R-squared of 100% for an overfit model

It appears that the model accounts for all of the variation. However, we know that the random predictors do not have any relationship to the random response! We are just fitting the random variability.

The predicted R-squared doesn’t have to be negative to indicate an overfit model. If you see the predicted R-squared start to fall as you add predictors, even if they’re significant, you should begin to worry about overfitting the model.

ll data contain a natural amount of variability that is unexplainable. Unfortunately, R-squared doesn’t respect this natural ceiling. Chasing a high R-squared value can push us to include too many predictors in an attempt to explain the unexplainable.

In these cases, you can achieve a higher R-squared value, but at the cost of misleading results, reduced precision, and a lessened ability to make predictions.

Both adjusted R-squared and predicted R-square provide information that helps you assess the number of predictors in your model:

Use the adjusted R-square to compare models with different numbers of predictors
Use the predicted R-square to determine how well the model predicts new observations and whether the model is too complicated

11) Differentiate between Ridge and Lasso Regression?

Lasso, Ridge and ElasticNet are all part of the Linear Regression family where the x (input) and y (output) are assumed to have a linear relationship. In sklearn, LinearRegression refers to the most ordinary least square linear regression method without regularization (penalty on weights) . The main difference among them is whether the model is penalized for its weights. For the rest of the post, I am going to talk about them in the context of scikit-learn library.

Linear regression (in scikit-learn) is the most basic form, where the model is not penalized for its choice of weights, at all. That means, during the training stage, if the model feels like one particular feature is particularly important, the model may place a large weight to the feature. This sometimes leads to overfitting in small datasets. Hence, following methods are invented.

Lasso is a modification of linear regression, where the model is penalized for the sum of absolute values of the weights. Thus, the absolute values of weight will be (in general) reduced, and many will tend to be zeros.

Ridge takes a step further and penalizes the model for the sum of squared value of the weights. Thus, the weights not only tend to have smaller absolute values, but also really tend to penalize the extremes of the weights, resulting in a group of weights that are more evenly distributed.

12) What is VIF? What is the suitable value of a VIF for a feature to be included in a regression modelling?

A variance inflation factor (VIF) is a measure of the amount of multicollinearity in regression analysis. Multicollinearity exists when there is a correlation between multiple independent variables in a multiple regression model. This can adversely affect the regression results. Thus, the variance inflation factor can estimate how much the variance of a regression coefficient is inflated due to multicollinearity.

KEY TAKEAWAYS
A variance inflation factor (VIF) provides a measure of multicollinearity among the independent variables in a multiple regression model.
Detecting multicollinearity is important because while multicollinearity does not reduce the explanatory power of the model, it does reduce the statistical significance of the independent variables.
A large VIF on an independent variable indicates a highly collinear relationship to the other variables that should be considered or adjusted for in the structure of the model and selection of independent variables.

A variance inflation factor(VIF) detects multicollinearity in regression analysis. Multicollinearity is when there's correlation between predictors (i.e. independent variables) in a model; it's presence can adversely affect your regression results. The VIF estimates how much the variance of a regression coefficient is inflated due to multicollinearity in the model.

VIFs are usually calculated by software, as part of regression analysis. You'll see a VIF column as part of the output. VIFs are calculated by taking a predictor, and regressing it against every other predictor in the model. This gives you the R-squared values, which can then be plugged into the VIF formula. "i" is the predictor you're looking at (e.g. x1 or x2):
variance inflation factor

Interpreting the Variance Inflation Factor
Variance inflation factors range from 1 upwards. The numerical value for VIF tells you (in decimal form) what percentage the variance (i.e. the standard error squared) is inflated for each coefficient. For example, a VIF of 1.9 tells you that the variance of a particular coefficient is 90% bigger than what you would expect if there was no multicollinearity — if there was no correlation with other predictors.
A rule of thumb for interpreting the variance inflation factor:

1 = not correlated.

$$\text{VIF} = \frac{1}{1 - R_i^2}$$

Between 1 and 5 = moderately correlated.
Greater than 5 = highly correlated.
Exactly how large a VIF has to be before it causes issues is a subject of debate. What is known is that the more your VIF increases, the less reliable your regression results are going to be. In general, a VIF above 10 indicates high correlation and is cause for concern. Some authors suggest a more conservative level of 2.5 or above.

Sometimes a high VIF is no cause for concern at all. For example, you can get a high VIF by including products or powers from other variables in your regression, like x and x2. If you have high VIFs for dummy variables representing nominal variables with three or more categories, those are usually not a problem.

13) Why do we need to scale the data before feeding it to the train the model?

Scaling of the data comes under the set of steps of data pre-processing when we are performing machine learning algorithms in the data set. As we know most of the supervised and unsupervised learning methods make decisions according to the data sets applied to them and often the algorithms calculate the distance between the data points to make better inferences out of the data.

In real life, if we take an example of purchasing apples from a bunch of apples, we go close to the shop, examine various apples and pick various apples of the same attributes. Because we have learned about the attributes of apples and we know which are better and which are not good also we know which attributes can be compromised and which can not. So if most of the apples consist of pretty similar attributes we will take less time in the selection of the apples which directly affect the time of purchasing taken by us. The moral of the example is if the apples every apple in the shop is good we will take less time to purchase or if the apples are not good enough we will take more time in the selection process which means that if the values of attributes are closer we will work faster and the chances of selecting good apples also strong.

Similarly in the machine learning algorithms if the values of the features are closer to each other there are chances for the algorithm to get trained well and faster instead of the data set where the data points or features values have high differences with each other will take more time to understand the data and the accuracy will be lower.

So if the data in any conditions has data points far from each other, scaling is a technique to make them closer to each other or in simpler words, we can say that the scaling is used for making data points generalized so that the distance between them will be lower.

As we know, most of the machine learning models learn from the data by the time the learning model maps the data points from input to output. And the distribution of the data points can be different for every feature of the data. Larger differences between the data points of input variables increase the uncertainty in the results of the model.

The machine learning models provide weights to the input variables according to their data points and inferences for output. In that case, if the difference between the data points is so high, the model will need to provide the larger weight to the points and in final results, the model with a large weight value is often unstable. This means the model can produce poor results or can perform poorly during learning.

I am not saying that all the algorithms will face this problem but most of the basic algorithms like linear and logistic regression, artificial neural networks, clustering algorithms with k value etc face the effect of the difference in scale for input variables.

Scaling the target value is a good idea in regression modelling; scaling of the data makes it easy for a model to learn and understand the problem. In the case of neural networks, an independent variable with a spread of values may result in a large loss in training and testing and cause the learning process to be unstable.

Normalization and Standardization are the two main methods for the scaling of the data. Which are widely used in the algorithms where scaling is required. Both of them can be implemented by the scikit-learn libraries preprocess package.

14) What are the different metrics which are used to check the goodness of fit in linear regression?

Model evaluation is very important in data science. It helps you to understand the performance of your model and makes it easy to present your model to other people. There are many different evaluation metrics out there but only some of them are suitable to be used for regression. This article will cover the different metrics for the regression model and the difference between them. Hopefully, after you read this post, you are clear on which metrics to apply to your future regression model.

Every time when I tell my friends: “Hey, I have built a machine learning model to predict XXX.” Their first reaction would be: “Cool, so what is the accuracy of your model prediction?” Well, unlike classification, accuracy in a regression model is slightly harder to illustrate. It is impossible for you to predict the exact value but rather how close your prediction is against the real value.

There are 3 main metrics for model evaluation in regression:
1. R Square/Adjusted R Square

2. Mean Square Error(MSE)/Root Mean Square Error(RMSE)

3. Mean Absolute Error(MAE)

R Square/Adjusted R Square
R Square measures how much variability in dependent variable can be explained by the model. It is the square of the Correlation Coefficient(R) and that is why it is called R Square.

R square formula
R Square is calculated by the sum of squared of prediction error divided by the total sum of the square which replaces the calculated prediction with mean. R Square value is between 0 to 1 and a bigger value indicates a better fit between prediction and actual value.

R Square is a good measure to determine how well the model fits the dependent variables. However, it does not take into consideration of overfitting problem. If your regression model has many independent variables, because the model is too complicated, it may fit very well to the training data but performs badly for testing data. That is why Adjusted R Square is introduced because it will penalize additional independent variables added to the model and adjust the metric to prevent overfitting issues.

```
#Example on R_Square and Adjusted R Square
import statsmodels.api as sm
X_addC = sm.add_constant(X)
result = sm.OLS(Y, X_addC).fit()
print(result.rsquared, result.rsquared_adj)
# 0.79180307318 0.790545085707
```
In Python, you can calculate R Square using Statsmodel or Sklearn Package

From the sample model, we can interpret that around 79% of dependent variability can be explained by the model, and adjusted R Square is roughly the same as R Square meaning the model is quite robust.

Mean Square Error(MSE)/Root Mean Square Error(RMSE)
While R Square is a relative measure of how well the model fits dependent variables, Mean Square Error is an absolute measure of the goodness for the fit.

Mean Square Error formula
MSE is calculated by the sum of square of prediction error which is real output minus predicted output and then divide by the number of data points. It gives you an absolute number on how much your predicted results deviate from the actual number. You cannot interpret many insights from one single result but it gives you a real number to compare against other model results and help you select the best regression model.

Root Mean Square Error(RMSE) is the square root of MSE. It is used more commonly than MSE because firstly sometimes MSE value can be too big to compare easily. Secondly, MSE is calculated by the square of error, and thus square root brings it back to the same level of prediction error and makes it easier for interpretation.

```
from sklearn.metrics import mean_squared_error
import math
print(mean_squared_error(Y_test, Y_predicted))
print(math.sqrt(mean_squared_error(Y_test, Y_predicted)))
# MSE: 2017904593.23
# RMSE: 44921.092965684235
```
MSE can be calculated in Python using Sklearn Package

Mean Absolute Error(MAE)
Mean Absolute Error(MAE) is similar to Mean Square Error(MSE). However, instead of the sum of square of error in MSE, MAE is taking the sum of the absolute value of error.

Mean Absolute Error formula
Compare to MSE or RMSE, MAE is a more direct representation of sum of error terms. MSE gives larger penalization to big prediction error by square it while MAE treats all errors the same.

```
from sklearn.metrics import mean_absolute_error
print(mean_absolute_error(Y_test, Y_predicted))
#MAE: 26745.1109986
```
MAE can be calculated in Python using Sklearn Package

Overall Recommendation/Conclusion
R Square/Adjusted R Square is better used to explain the model to other people because you can explain the number as a percentage of the output variability. MSE, RMSE, or MAE are better be used to compare performance between different regression models. Personally, I would prefer using RMSE and I think Kaggle also uses it to assess the submission. However, it makes total sense to use MSE if the value is not too big and MAE if you do not want to penalize large prediction errors.

Adjusted R square is the only metric here that considers the overfitting problem. R Square has a direct library in Python to calculate but I did not find a direct library to calculate Adjusted R square except using the statsmodel results. If you really want to calculate Adjusted R Square, you can use stats model or use its mathematic formula directly.

15)From the following confusion matrix calculate sensitivity, specificity, precision, recall and accuracy.

The million-dollar question – what, after all, is a confusion matrix?

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:

Confusion matrix

Let's decipher the matrix:

The target variable has two values: Positive or Negative
The columns represent the actual values of the target variable
The rows represent the predicted values of the target variable
But wait – what's TP, FP, FN and TN here? That's the crucial part of a confusion matrix. Let's understand each term below.

Understanding True Positive, True Negative, False Positive and False Negative in a Confusion Matrix
True Positive (TP)

The predicted value matches the actual value
The actual value was positive and the model predicted a positive value
True Negative (TN)

The predicted value matches the actual value
The actual value was negative and the model predicted a negative value
False Positive (FP) – Type 1 error

The predicted value was falsely predicted
The actual value was negative but the model predicted a positive value
Also known as the Type 1 error
False Negative (FN) – Type 2 error

The predicted value was falsely predicted
The actual value was positive but the model predicted a negative value
Also known as the Type 2 error
Let me give you an example to better understand this. Suppose we had a classification dataset with 1000 data points. We fit a classifier on it and get the below confusion matrix:

Confusion matrix example

The different values of the Confusion matrix would be as follows:

True Positive (TP) = 560; meaning 560 positive class data points were correctly classified by the model

True Negative (TN) = 330; meaning 330 negative class data points were correctly classified by the model
False Positive (FP) = 60; meaning 60 negative class data points were incorrectly classified as belonging to the positive class by the model
False Negative (FN) = 50; meaning 50 positive class data points were incorrectly classified as belonging to the negative class by the model
This turned out to be a pretty decent classifier for our dataset considering the relatively larger number of true positive and true negative values.

Remember the Type 1 and Type 2 errors. Interviewers love to ask the difference between these two! You can prepare for all this better from our Machine Learning Course Online

Let’s say you want to predict how many people are infected with a contagious virus in times before they show the symptoms, and isolate them from the healthy population (ringing any bells, yet? 😷). The two values for our target variable would be: Sick and Not Sick.

Now, you must be wondering – why do we need a confusion matrix when we have our all-weather friend – Accuracy? Well, let’s see where accuracy falters.

Our dataset is an example of an imbalanced dataset. There are 947 data points for the negative class and 3 data points for the positive class. This is how we’ll calculate the accuracy:Equation_Accuracy

Let’s see how our model performed:

Dataset

The total outcome values are:

TP = 30, TN = 930, FP = 30, FN = 10

So, the accuracy for our model turns out to be:

Confusion Marix Accuracy

96%! Not bad!

But it is giving the wrong idea about the result. Think about it.

Our model is saying “I can predict sick people 96% of the time”. However, it is doing the opposite. It is predicting the people who will not get sick with 96% accuracy while the sick are spreading the virus!

Do you think this is a correct metric for our model given the seriousness of the issue? Shouldn’t we be measuring how many positive cases we can predict correctly to arrest the spread of the contagious virus? Or maybe, out of the correctly predicted cases, how many are positive cases to check the reliability of our model?

This is where we come across the dual concept of Precision and Recall.

Precision vs. Recall
Precision tells us how many of the correctly predicted cases actually turned out to be positive.

Confusion_Matrix_Precision_Recall

50% percent of the correctly predicted cases turned out to be positive cases. Whereas 75% of the positives were successfully predicted by our model. Awesome!

Precision is a useful metric in cases where False Positive is a higher concern than False Negatives.

Precision is important in music or video recommendation systems, e-commerce websites, etc. Wrong results could lead to customer churn and be harmful to the business.

Recall is a useful metric in cases where False Negative trumps False Positive.

Recall is important in medical cases where it doesn’t matter whether we raise a false alarm but the actual positive cases should not go undetected!

In our example, Recall would be a better metric because we don’t want to accidentally discharge an infected person and let them mix with the healthy population thereby spreading the contagious virus. Now you can understand why accuracy was a bad metric for our model.

But there will be cases where there is no clear distinction between whether Precision is more important or Recall. What should we do in those cases? We combine them!

F1-Score
In practice, when we try to increase the precision of our model, the recall goes down, and vice-versa. The F1-score captures both the trends in a single value:

Confusion Matrix F1-score

F1-score is a harmonic mean of Precision and Recall, and so it gives a combined idea about these two metrics. It is maximum when Precision is equal to Recall.

But there is a catch here. The interpretability of the F1-score is poor. This means that we don’t know what our classifier is maximizing – precision or recall? So, we use it in combination with other evaluation metrics which gives us a complete picture of the result.