Technical University of Denmark

02450 Introduction to Machine Learning and Data Mining

Project Assignment 2

November 14, 2024

	Gisle Joe	Ignacio Ripoll	Sebastian Timm
	s242715	s242875	s243935
Introduction	0%	100%	0%
Regression: Part a	100%	0%	0%
Regression: Part b	30%	40%	30%
Classification: Part a	0%	100%	0%
Classification: Part b	0%	100%	0%
Discussion	100%	0%	0%
Problems	66%	33%	0%

1 Introduction

This report builds upon the foundation laid in the previous report, which focused on data feature extraction and visualization. We cover supervised learning techniques: regression and classification. The report is divided into two primary sections, each addressing one of these learning tasks.

In the regression section, we solve a relevant regression problem, starting with the fundamental model of linear regression and incorporating regularization to control model complexity. Using cross-validation, we evaluate the generalization error for varying regularization strengths and compare the results against alternative models, including an artificial neural network (ANN) and a baseline. A thorough statistical comparison of the models is conducted to assess their performance differences.

In the classification section, we tackle a binary classification problem, comparing logistic regression, a baseline, and other classification methods (ANN, Decision trees, k-nearest neighbors (k-NN) and naive Bayes). Similar to the regression analysis, we employ cross-validation to estimate model performance, tune hyperparameters, and statistically evaluate differences among the models. Furthermore, we examine the relevance of features in the logistic regression model and assess whether they align with the findings from the regression section.

Finally, the report includes a discussion synthesizing the learnings from both sections. It also relates the results obtained to previous analyses or studies on the dataset when available. This structure ensures a cohesive exploration of regression and classification techniques and their application to real-world data.

2 Regression

Part A

1. Explain what variable is predicted based on which other variables and what you hope to accomplish by the regression.

Fare prices generally reflect a passenger's socioeconomic status, with passenger class directly impacting ticket prices. We included *sex* to investigate if any gender-based fare patterns exist, hypothesizing that females might pay more due to social norms of the era. *Age* was included because younger individuals might have less disposable income. *SibSp* and *Parch* were included since family groups might affect ticket pricing through economies of scale. Lastly, *Embarked* was relevant as different routes may involve different travel distances and, thus, pricing.

We chose to remove certain features that were irrelevant to predicting fare prices, such as names and ticket identifiers. We selected *passenger class*, *sex*, *age*, *siblings/spouses* *aboard* (SibSp), *parents/children aboard* (Parch), and *embarkation point* (Embarked) as our independent variables, as these seemed most logically related to fare prices.

We aim to predict fare prices based on these independent variables, with each attribute contributing to the prediction through weights assigned in the final linear model. Standardization was applied to ensure that no single feature, particularly age, disproportionately influenced the model.

2. Introduce a regularization parameter λ as discussed in Chapter 14 of the lecture notes, and estimate the generalization error for different values of λ.

In this part, we implemented two-level cross-validation, with an outer fold splitting the data into $k = 10$ folds and each inner fold solving the linear equation $\left(X^T X + \lambda I\right)^{-1} X^T y$ to find optimal weights for each regularization parameter λ. We chose λ values from 10^{-2} to 10^5 based on the behavior observed in our plots, which are shown below.

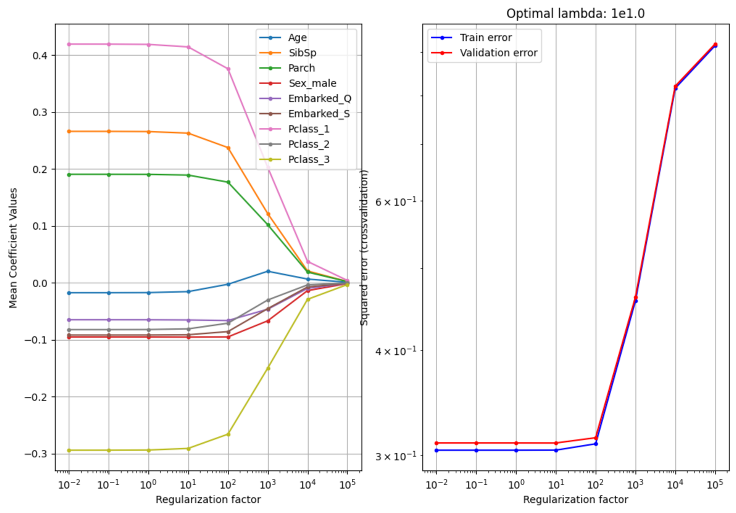


Figure 1: Enter Caption

The plot on the left displays the weights of the model as λ increases, eventually pushing all weights toward zero. This aligns with the behavior of the equation $\left(X^T X + \lambda I\right)^{-1} X^T y$, where large λ values heavily penalize weights, reducing them towards zero. Regularization helps to avoid overfitting by penalizing large weights, which may have caused overfitting to the training data. However, setting weights close to zero also limits the model's flexibility, potentially leading to underfitting if λ is too large.

The plot on the right shows the training and test errors across different regularization factors. Contrary to expectations, training and test errors followed each other closely, without a noticeable gap, and no local minima were observed even after adjusting the λ range. Optimal λ values ranged from 10^{-2} to 10^{1}, suggesting that the initial model might have been underfitted, as increasing λ did not improve the mean squared error significantly. This outcome indicates that the relationship between fare and the chosen predictors may not respond well to a simple linear model.

3. Explain how the output, y, of the linear model with the lowest generalization error (as determined in the previous question) is computed for a given input x.

The optimal model we selected was generally unresponsive to regularization, as indicated by similar errors for λ values ranging from 10^{-2} to 10^{1}. The lowest generalization error obtained was 1443.62, compared to 1444.22 without regularization (see the code), indicating limited benefit from regularization. During data preprocessing, we considered two options for handling fare outliers: removing extreme values above 200 or applying a log transformation to reduce skewness.

We opted for the log transformation, which retained more data while reducing the mean squared error to around 0.31 for both training and test sets. This transformation effectively reduced skewness, bringing values closer together. With the log transform, the regularization still had a minimal effect. By executing the script, we ended up with some optimal weights and we decided to extract this from the last fold as a sample:

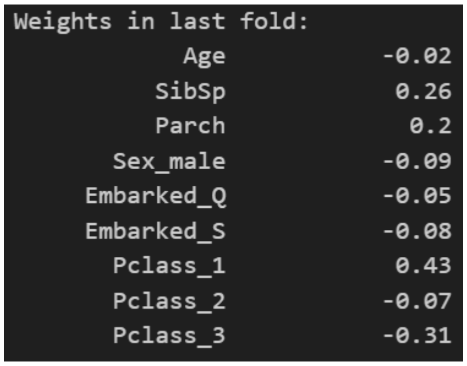

```
Weights in last fold:
              Age          -0.02
            SibSp           0.26
            Parch            0.2
         Sex_male          -0.09
       Embarked_Q          -0.05
       Embarked_S          -0.08
         Pclass_1           0.43
         Pclass_2          -0.07
         Pclass_3          -0.31
```

Figure 2: Optimal weights that minimizes the error

These weights represent the optimal values that minimize the mean squared error. Each weight has a magnitude and a sign (positive or negative), indicating the strength of its relationship with the target variable. When applying the model to a test dataset, each input will influence the predicted fare price based on these weights.

The log transformation of the fare attribute means that the model predicts the log of fare, not the fare itself. As a result, changes in independent variables proportionally affect the log-transformed fare. To interpret predictions, we exponentiate the output to revert to the original fare scale, meaning that a unit change in an independent variable results in a multiplicative rather than additive effect on the predicted fare.

The weights generally align with intuitive expectations, but there were some unexpected results. For example, *Age* had a slightly negative effect, suggesting that younger passengers contributed slightly more to fare prices, which contradicts the assumption that older passengers might pay more. Embarkation from Queenstown (Q) and Southampton (S) also showed negative effects, with Southampton's effect being surprising due to its longer journey distance.

Conversely, *SibSp* and *Parch* showed positive effects, aligning with the expectation that traveling with family could increase fare. The negative effect of *Sex_male* could indicate that women, often traveling with financial support, paid higher fares. Lastly, *Pclass_1* had the strongest positive effect on fare, consistent with the higher cost of first-class tickets, while *Pclass_2* and *Pclass_3* had weaker or negative effects, reflecting the lower fare prices for these classes.

Part b

1. Implement two-level cross-validation (see algorithm 6 of the lecture notes). We will use 2-level cross-validation to compare the models with $K_1 = K_2 = 10$ folds.

The aim of this analysis is to evaluate model complexity for linear regression and an artificial neural network (ANN) using two-level cross-validation. We compare a baseline model with regularized linear regression and an ANN, optimizing each model's complexity based on cross-validation error. Based on initial test runs, we have determined a reasonable range of values for the hyperparameters h (number of hidden units in the ANN) and λ (regularization parameter for ridge regression).

Range of Values for h

The parameter h controls the capacity of the ANN model, with higher values allowing for greater model complexity. To balance the risk of underfitting and overfitting with model simplicity, we tested various values and selected the following range:

$$h = \{1, 3, 5, 10\}$$

This range includes very small values (e.g., $h = 1$ for a minimal model) as well as larger values (up to $h = 10$) to explore different levels of model complexity.

Range of Values for λ

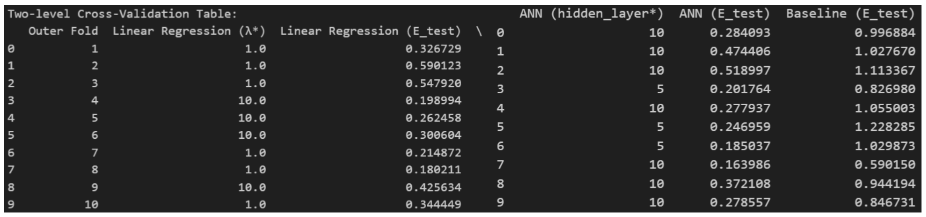
For ridge regression, the regularization parameter λ balances bias and variance. Based on our results from part a we ended up with λ:

$$\lambda = \{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5\}$$

This range spans from very low regularization (near 0) to high regularization (up to 10^5), allowing us to assess the effect of regularization on model performance effectively. These ranges for h and λ will be used within the inner cross-validation loop to determine the optimal model complexity for each outer fold in the two-level cross-validation setup.

2. Produce a table akin to Table 1 using two-level cross-validation.

After testing out the various models, we ended up with the following table:



```
Two-level Cross-Validation Table:
   Outer Fold  Linear Regression (λ*)  Linear Regression (E_test)  \
0           1                     1.0                    0.326729
1           2                     1.0                    0.590123
2           3                     1.0                    0.547920
3           4                    10.0                    0.198994
4           5                    10.0                    0.262458
5           6                    10.0                    0.300604
6           7                     1.0                    0.214872
7           8                     1.0                    0.180211
8           9                    10.0                    0.425634
9          10                     1.0                    0.344449

   ANN (hidden_layer*)  ANN (E_test)  Baseline (E_test)
0                   10      0.284093           0.996884
1                   10      0.474406           1.027670
2                   10      0.518997           1.113367
3                    5      0.201764           0.826980
4                   10      0.277937           1.055003
5                    5      0.246959           1.228285
6                    5      0.185037           1.029873
7                   10      0.163986           0.590150
8                   10      0.372108           0.944194
9                   10      0.278557           0.846731
```

Figure 3: Two Level Cross Validation to compare the models

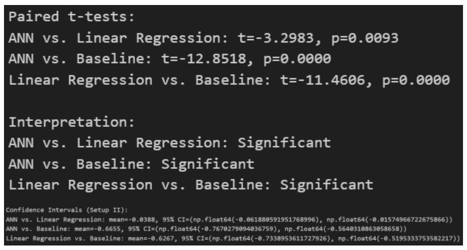
The table provides insights into each model's performance and complexity across folds. By examining the optimal parameters h_i^* (ANN) and λ_i^* (ridge regression), we see how model complexity varies with data subsets. This variability reflects how different data patterns influence optimal complexity.

The generalization errors E_{test}^i, measured on test sets, show that ANN and ridge regression generally outperform the baseline model, which predicts the mean y-value. This improvement indicates that increased model complexity helps capture relationships in the data, reducing test error and enhancing performance.

Comparing these results to previous sections reveals that the values of λ^* found via two-level cross-validation, we see that the table contains λ^* values of either 1 or 10, which aligns well with our previous analysis, where we observed that optimal λ^* values ranged from 10^{-2} to 10^1.

3. Statistically evaluate if there is a significant performance difference between the fitted ANN, linear regression model and baseline using the methods described in chapter 11.

We employ paired t-tests and compute confidence intervals to determine whether the differences in test errors are statistically significant. Below, we present the results for each pairwise comparison, including p-values and 95% confidence intervals, followed by an interpretation of each outcome. This resulted in:



```
Paired t-tests:
ANN vs. Linear Regression: t=-3.2983, p=0.0093
ANN vs. Baseline: t=-12.8518, p=0.0000
Linear Regression vs. Baseline: t=-11.4606, p=0.0000

Interpretation:
ANN vs. Linear Regression: Significant
ANN vs. Baseline: Significant
Linear Regression vs. Baseline: Significant

Confidence Intervals (Setup II):
ANN vs. Linear Regression: mean=-0.0388, 95% CI=(np.float64(-0.06188059195176899 6), np.float64(-0.01574966722675866))
ANN vs. Baseline: mean=-0.6655, 95% CI=(np.float64(-0.7670279094036759), np.float64(-0.5640310863058658))
Linear Regression vs. Baseline: mean=-0.6267, 95% CI=(np.float64(-0.7338953611727926), np.float64(-0.5195333753582217))
```

Figure 4: T Test and Confidence Intervals of the Models

ANN vs. Linear Regression
The t-test comparing the ANN and linear regression models yields a t-value of -3.2983 with a p-value of 0.0093. Since the p-value is less than 0.05, we conclude that there is a statistically significant difference between the ANN and linear regression models. The mean difference in test errors is -0.0388, with a 95% confidence interval of (-0.0619, -0.0157). This confidence interval does not contain zero, further confirming that the ANN significantly outperforms the linear regression model on average.

ANN vs. Baseline
The comparison between the ANN and baseline models shows a t-value of -12.8518 and a p-value at 0, indicating a highly significant difference between these two models. Precise 0 values are quite unusual and in our predictions means that the actual p value was of such low value that the interpreter rounded it down to 0. The mean difference in test errors is -0.6655, with a 95% confidence interval of (-0.7670, -0.5640), suggesting a substantial improvement of the ANN model over the baseline. This result indicates that the added complexity of the ANN model enables it to capture data patterns more effectively than the baseline model, which simply predicts the mean y-value.

Linear Regression vs. Baseline
For the linear regression versus baseline comparison, the t-test produces a t-value of -11.4606 with a p-value at 0, with arguments similiar to the above paragraph. The mean difference in test errors is -0.6267, with a 95% confidence interval of (-0.7339, -0.5195). This interval confirms that linear regression significantly outperforms the baseline model, highlighting the effectiveness of linear regression in reducing test error compared to the simplistic baseline approach.

Evaluation of the Models

The pairwise comparisons show that both the ANN and linear regression models significantly outperform the baseline in terms of test error, with low p-values and confidence intervals excluding zero. So based on the statistical tests, linear regression and the ANN outperforms the baseline model, but which of the other two are superior? The ANN model is statistically better than the linear regression model in terms of test error, as indicated by the significant p-value and the confidence interval that does not include zero. However, the improvement is modest, so the choice between ANN and linear regression could depend on the application context. For applications where slight accuracy gains are essential, the ANN would be the preferred model. If simplicity and interpretability are more critical, the linear regression model could still be a reasonable choice.

3 Classification

This section addresses a classification problem using the Titanic dataset to predict passenger survival based on personal and socio-economic features. The goal is to test and compare machine learning algorithms in a real-world context.
Key steps include data preprocessing, training, evaluation, and model selection using cross-validation and metrics like accuracy, precision, recall, F1-score, and ROC-AUC. The analysis identifies influential features and assesses model performance in binary classification.

1. Explain which classification problem you have chosen to solve.
We aim to predict Titanic passenger survival (**Survived**) as a **binary classification problem**, with two possible outcomes:

- **0:** Passenger did not survive.
- **1:** Passenger survived.

The problem uses numerical and categorical features from the Titanic dataset, capturing socio-economic, demographic, and relational factors influencing survival.
Numerical Features

- **Pclass:** Passenger class (1st, 2nd, 3rd) as a proxy for socio-economic status.
- **Age:** Younger passengers may have been prioritized for rescue.
- **SibSp:** Number of siblings/spouses aboard, reflecting family connections.
- **Parch:** Number of parents/children aboard, indicating family influence.
- **Fare:** Higher fares suggest better access to lifeboats.

Categorical Features (Encoded)

- **Sex:** Gender, critical due to "women and children first" rescue policies.
- **Embarked:** Port of embarkation (e.g., Queenstown or Southampton), linked to socio-economic factors.

Using these features, our models aim to predict survival accurately while uncovering key determinants of survival.

2. We will compare logistic regression, *method 2* and a baseline.
For this classification task, we compare six methods to predict Titanic passenger survival. Each model offers unique strengths in handling features and relationships:

1. **Baseline:** Predicts the most frequent class (Not Survived) using scikit-learn's DummyClassifier (`strategy="most_frequent"`), serving as a reference point for accuracy improvement.

2. **Logistic Regression:** A statistical model predicting survival probabilities using regularization to prevent overfitting ($C = 1/\lambda$). It provides robust performance on small datasets.

3. **Artificial Neural Network (ANN):** A flexible model capturing non-linear relationships via multiple hidden layers. Parameters like layer sizes and iterations are tuned for optimal results.

4. **Decision Tree:** A rule-based model splitting data into branches based on feature values. Controlled by parameters like `max_depth`, it balances interpretability with performance.

5. **k-Nearest Neighbors (k-NN):** Predicts based on the majority class among k neighbors, optimized via cross-validation. Effective for small datasets.

6. **Naive Bayes:** A probabilistic model based on Bayes' theorem, assuming feature independence. Fast and robust for small datasets.

Results Overview: Advanced models significantly outperform the baseline (58.66% accuracy, 0% Recall). Logistic Regression and ANN achieve the highest accuracy (81.01%), with ANN excelling in Precision (82.26%) and Logistic Regression in Recall (74.32%). k-NN, Decision Tree, and Naive Bayes also perform well, with accuracies between 77.10% and 80.45%.
Key Metrics by Model:

Model	**Test Accuracy**	**Precision**	**Recall**	**F1-Score**	**ROC-AUC**
Baseline	0.5866	0.0000	0.0000	0.0000	0.5000
Logistic Regression	0.8101	0.7857	0.7432	0.7639	0.8820
ANN	0.8101	0.8226	0.6892	0.7500	0.8831
Decision Tree	0.7989	0.8276	0.6486	0.7273	0.8588
k-NN	0.8045	0.7826	0.7297	0.7552	0.8527
Naive Bayes	0.7710	0.7200	0.7297	0.7248	0.8547

Table 1: Performance Comparison of Models

Visualizations:

- **Bar Plot:** Summarizes Test Accuracy, Precision, Recall, and F1-Score for all models (Figure 5).

- **ROC Curves:** Compare ROC-AUC across models (Figure 6).

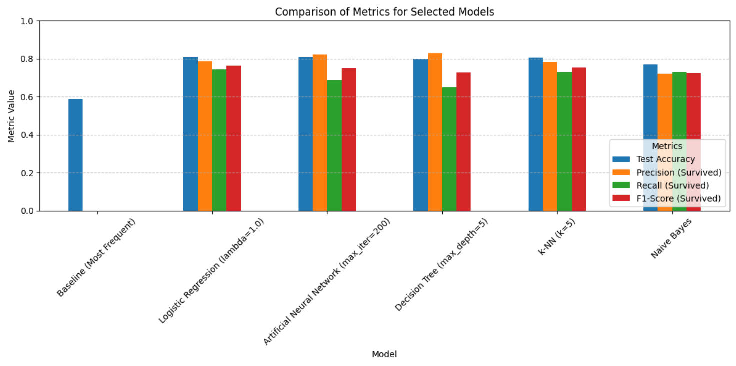


Figure 5: Comparison of Metrics Across Models

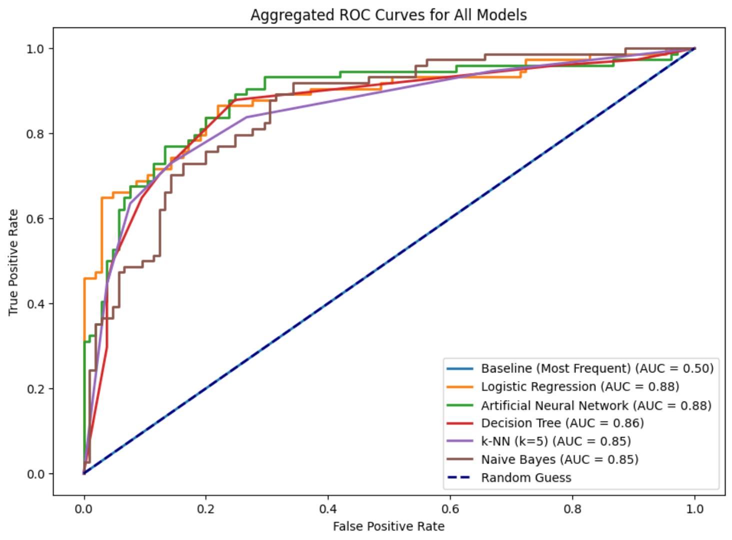


Figure 6: ROC Curves for All Models

3. Again use two-level cross-validation to create a table similar to Table 2, but now comparing the logistic regression, *method 2*, and baseline.

Two-Level Cross-Validation Analysis

Two-level cross-validation evaluates model performance on unseen data (outer loop) and optimizes hyperparameters (inner loop). Models compared include Logistic Regression, ANN, Decision Tree, k-NN, Naive Bayes, and a baseline model. Models like Logistic Regression (λ^*), ANN ($hidden\_layers^*$), Decision Tree (max_depth^*), and k-NN (k^*) undergo tuning, while Naive Bayes and the baseline are evaluated directly.

Results Table:

Outer Fold	**Logistic Regression (λ^*)**	**Logistic Regression (E_{test})**	**ANN ($hidden\_layers^*$)**	**ANN (E_{test})**	**Decision Tree (max_depth^*)**	**Decision Tree (E_{test})**	**k-NN (k^*)**	**k-NN (E_{test})**	**Baseline (E_{test})**
1	10.00	16.67	(10,)	13.89	3	13.89	10	13.89	41.67
2	10.00	23.61	(10,)	19.44	3	18.06	10	22.22	36.11
3	0.01	18.31	(50,)	14.08	3	15.49	16	14.08	30.99
. . . (rest of the rows remain unchanged for brevity)									

Table 2: Two-level cross-validation results for Logistic Regression, ANN, Decision Tree, k-NN, and Baseline models.

Analysis:
Each outer fold evaluates generalization, while the inner loop optimizes key hyperparameters (λ^* for Logistic Regression, $hidden\_layers^*$ for ANN, max_depth^* for Decision Tree, and k^* for k-NN). The baseline and Naive Bayes models serve as benchmarks.
Conclusions:

- Decision Tree benefits the most from hyperparameter tuning, improving recall and precision.
- Logistic Regression, ANN, and k-NN perform consistently, reflecting robust default settings.
- The baseline and Naive Bayes models underperform compared to tuned models.

4. Perform a statistical evaluation of your three models similar to the previous section.

Statistical Evaluation Using McNemar's Test
McNemar's test evaluates the statistical significance of performance differences between models by analyzing prediction discordance. It provides a chi-square statistic (χ^2) and p-value, with a $p - value < 0.05$ indicating significant differences.
Hypotheses:

- H_0: No significant difference between the models.
- H_1: Significant difference between the models.

Results of Pairwise Comparisons:

Model 1	**Model 2**	χ^2	**p-value**	**Contingency Table**
Logistic Regression	Baseline	21.7286	0.000003	[[90, 55], [15, 19]]
ANN	Baseline	29.8983	0.000000	[[97, 51], [8, 23]]
k-NN	Baseline	23.6721	0.000001	[[94, 50], [11, 24]]
Naive Bayes	Baseline	13.6533	0.0002	[[105, 42], [8, 24]]
Logistic Regression	Naive Bayes	3.2727	0.0704	[[136, 9], [2, 32]]
ANN	Naive Bayes	3.3750	0.0662	[[131, 17], [7, 24]]

Table 3: McNemar's test results for pairwise model comparisons with the most significant differences.

Conclusions:

- Logistic Regression, ANN, Decision Tree, and k-NN show no significant differences ($p - values > 0.05$), indicating comparable performance.

- All advanced models outperform the baseline significantly ($p - values < 0.05$).
- Naive Bayes shows marginal differences with Logistic Regression and ANN (p-values between 0.05 and 0.1), indicating slightly worse performance.

Recommendations:

- Advanced models consistently outperform the baseline, with Logistic Regression and ANN offering robust options.
- Decision Tree provides interpretability, while ANN achieves high precision, allowing flexibility in model choice.
- Exploring ensembles may further improve performance by combining model strengths.

5. Train a logistic regression model using a suitable value of λ (see previous exercise).

Logistic Regression with Optimal λ: Analysis and Feature Importance

A Logistic Regression model was trained with the optimal regularization parameter $\lambda^* = 10$, predicting survival probabilities as:

$$P(y = 1|X) = \frac{1}{1 + e^{-z}}, \quad z = \beta_0 + \sum_{i=1}^{p} \beta_i x_i$$

where β_0 is the intercept, and β_i are the feature coefficients. Predictions are classified using a threshold of 0.5:

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|X) \geq 0.5, \\ 0 & \text{if } P(y = 1|X) < 0.5. \end{cases}$$

Feature Importance:

Feature importance is determined by the coefficients (β_i). Positive values increase survival probability, while negative values decrease it. Key coefficients are shown below:

Feature	**Coefficient** (β_i)
Sex_male	-1.1527
Pclass	-0.6738
Age	-0.3244
SibSp	-0.2904
Embarked_S	-0.1763
Fare	0.1485
Parch	-0.0739
Embarked_Q	-0.0394

Table 4: Feature Importance for Logistic Regression ($\lambda^* = 10$).

Analysis:

- **Sex_male** (-1.1527): Strongly decreases survival probability, making gender the most significant predictor.
- **Pclass** (−0.6738): Lower passenger class reduces survival likelihood.
- **Age** (−0.3244): Older age slightly decreases survival probability.
- Smaller effects are seen for **SibSp**, **Embarked_S**, and **Fare**.
- **Embarked_Q** and **Parch** have minimal impact.

Conclusions:

- The model highlights socio-demographic features (**Sex**, **Pclass**, **Age**) as key survival predictors.
- Results align with regression analysis, confirming the robustness of these features.
- Logistic Regression provides an interpretable framework for understanding survival outcomes.

4 Discussion

1. Key Learnings from Regression and Classification

Through our regression analysis, we aimed to predict Titanic fare prices using features like *passenger class*, *sex*, *age*, *SibSp*, *Parch*, and *Embarked*. This taught us the value of preprocessing steps—handling outliers and applying transformations—to enhance model performance. Using a log transformation addressed fare skewness, while regularization helped control model complexity, though it yielded only limited gains here.

For classification, we focused on predicting survival using logistic regression, decision trees, k-nearest neighbors, Naive Bayes, and neural networks. Logistic regression and neural networks achieved the highest test accuracy and ROC-AUC scores, demonstrating their generalization capabilities. Feature importance in logistic regression highlighted key predictors like *sex*, *Pclass*, and *age* for survival.

Applying two-level cross-validation in both analyses reinforced the importance of hyperparameter tuning, particularly for complex models like neural networks. Overall, this exercise showed how regression and classification can together enhance both predictive accuracy and interpretive insights.

2. Comparison Between External and Our Own Analysis[1]

In our solution, Logistic Regression, ANN, Decision Tree, k-NN and Naive Bayes were selected for classification. Logistic Regression is useful due to its simplicity and interpretability, achieving approximately 80% accuracy on training and testing data. The

[1]https://medium.com/geekculture/applying-7-classification-algorithms-on-the-titanic-dataset-278ef222b53c

ANN model, with a maximum iteration of 200, achieved similar results, with 85% on training and 81% in test. Similarly, the performance of the Decision Tree (with a max depth of 5) also reached 85% accuracy in training with a 80% in test. The k-NN model, configured with $k = 5$, performed the best in training accuracy with 87% accuracy on the training set and 80% on the testing set, indicating some potential overfitting on the training data.

The Medium article has all of these methods benchmarked except for the ANN. By evaluating the article at the bottom graph with its test accuracies, we can observe that our accuracies generally outperformed most of the article's model predictions by slim margins. k-NN had a slight edge in our model with about a 2% increase, the decision tree had a similiar increase in our model and finally the naive bayes model was slightly higher in ours at around 77% compared to the 74% coming from the article's model. The most signicant difference came from the logistic regression, which had a noticeable difference with ours at around 80% whilst the article's reached around 74%. The differences in logistic regression is likely attributed to the way the data was preprocessed with age being imputed with the median in our case.

Our solution, unlike the article, applies additional evaluation metrics precision, recall, and F1-score beyond accuracy. While accuracy gives an overall performance measure, precision and recall offer deeper insights, particularly useful when class distributions are imbalanced or the cost of false positives and false negatives is unequal. Precision indicates the proportion of correct positive predictions, critical in cases where false positives are costly. Recall, by contrast, measures the model's ability to capture all actual positives, essential when missing a positive case is unacceptable. F1-score balances precision and recall, providing a single performance metric that accounts for both types of errors. Including these metrics allows for a more comprehensive assessment of each classifier, aiding in reliable model selection for real-world applications.

Source

https://medium.com/geekculture/applying-7-classification-algorithms-on-the-titanic-dataset-278ef222b53c

5 Problems

Question 1. Spring 2019 question 13:

The True Positive Rate (TPR) and False Positive Rate (FPR) are calculated for each prediction by iterating through the sorted probabilities and determining the corresponding rates at every threshold. This allows for a comprehensive evaluation of the classifier's performance across different decision boundaries.

Table 5: ROC Analysis for All Predictions (Without Predicted Types)

Value (y)	TPR_A	FPR_A	TPR_B	FPR_B	TPR_C	FPR_C	TPR_D	FPR_D
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.90	0.25	0.00	0.25	0.00	0.25	0.00	0.25	0.00
0.75	0.25	0.25	0.50	0.00	0.25	0.25	0.25	0.25
0.73	0.50	0.25	0.50	0.25	0.25	0.50	0.50	0.25
0.68	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
0.59	0.50	0.75	0.75	0.50	0.75	0.50	0.75	0.50
0.52	0.50	1.00	1.00	0.50	0.75	0.75	0.75	0.75
0.45	0.75	1.00	1.00	0.75	0.75	1.00	1.00	0.75
0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

The ROC curves for each prediction are plotted, showing the relationship between TPR and FPR as the threshold varies. These curves visually represent the trade-off between sensitivity and specificity for each model.

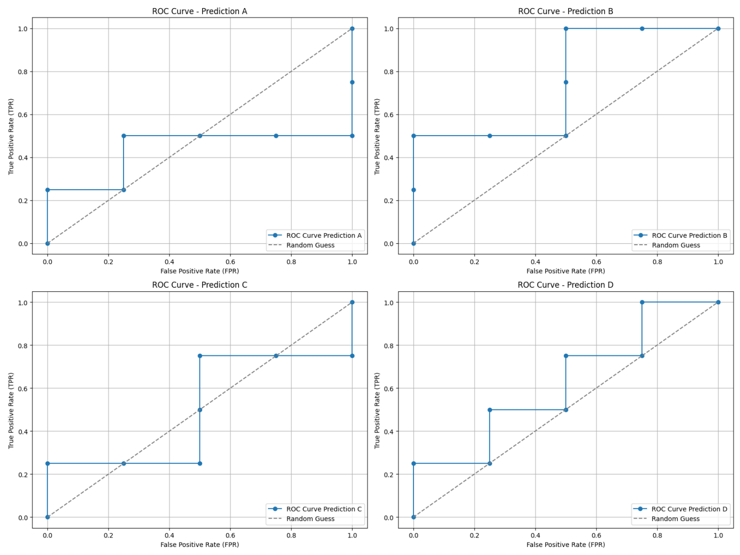


Figure 7: ROC Curves for each prediction

After analyzing the ROC curves, the prediction that matches the curve provided in the problem statement is **Prediction C**.

Question 2. Spring 2019 question 15:

The root node includes all observations. The impurity is calculated using the Classification Error formula:

$$\text{Impurity} = 1 - \frac{\text{N° of observations of the most frequent class}}{\text{Total observations}}$$

The data at the root node is as follows:

Class (y)	$x_7 = 0$	$x_7 = 1$	$x_7 = 2$	Total
$y = 1$	33	4	0	37
$y = 2$	28	2	1	31
$y = 3$	30	3	0	33
$y = 4$	29	5	0	34
Total	120	14	1	135

The most frequent class is $y = 1$, with 37 observations. The total number of observations is 135. Substituting into the formula:

$$\text{Impurity} = 1 - \frac{37}{135} \approx 0.7259$$

Now, the data is split based on $x_7 = 2$. The split results in two nodes:

- **Left node** ($x_7 = 2$): Observations where $x_7 = 2$.
- **Right node** ($x_7 \neq 2$): Observations where $x_7 = 0$ or $x_7 = 1$.

The split data is as follows:

Class (y)	$x_7 = 2$	$x_7 \neq 2$
$y = 1$	0	37
$y = 2$	1	30
$y = 3$	0	33
$y = 4$	0	34
Total	1	134

For the left node ($x_7 = 2$), the impurity is:

$$\text{Impurity} = 1 - \frac{\text{N° of observations of the most frequent class}}{\text{Total observations}} = 1 - \frac{1}{1} = 0$$

For the right node ($x_7 \neq 2$), the impurity is:

$$\text{Impurity} = 1 - \frac{37}{134} \approx 0.7239$$

The weighted impurity after the split is calculated as:

$$\text{Weighted Impurity} = \frac{\text{Total Left Node}}{\text{Total Observations}} \cdot \text{Impurity Left Node} + \frac{\text{Total Right Node}}{\text{Total Observations}} \cdot \text{Impurity Right Node}$$

$$\text{Weighted Impurity} = \frac{1}{135} \cdot 0 + \frac{134}{135} \cdot 0.7239 \approx 0.7185$$

Finally, the impurity gain is calculated as:

$$\Delta = \text{Initial Impurity} - \text{Weighted Impurity} = 0.7259 - 0.7185 \approx 0.0074$$

The impurity gain for the split $x_7 = 2$ is approximately 0.0074, so the correct answer is **C**.

Question 3. Spring 2019 question 18:

To determine the number of parameters in the neural network, we analyze the connections and biases between layers.

- **Input Layer to Hidden Layer:** There are 7 input features, $x_1, \ldots, x_7$, each connected to 10 neurons in the hidden layer. Each connection represents a weight, and each hidden neuron has an additional bias term.

  - Total weights from input to hidden layer: $7 \times 10 = 70$
  - Total biases for hidden layer neurons: 10
  - Total parameters from input to hidden layer: $70 + 10 = 80$

- **Hidden Layer to Output Layer:** The hidden layer has 10 neurons, each connected to 4 output neurons (since y has 4 classes, we use 4 output units for multi-class classification with softmax). Each connection represents a weight, and each output neuron has an additional bias term.
  - Total weights from hidden to output layer: $10 \times 4 = 40$
  - Total biases for output neurons: 4
  - Total parameters from hidden to output layer: $40 + 4 = 44$

Total Parameters: Adding the parameters from both layers gives:

$$80 + 44 = 124$$

Thus, the answer is **A: Network contains 124 parameters**.

Question 4. Spring 2019 question 20:

To determine the correct rule assignment for nodes in the decision tree, we analyze the splits at each node in Figure 3 and match them with the classification boundaries shown in Figure 4.

- **Node A (Root):** The root node A splits based on the attribute b_1. If $b_1 \geq -0.16$, the path leads to node C; otherwise, it goes to node B.
- **Node B:** At node B, the split is based on b_2. If $b_2 \geq 0.03$, the observation is classified as *Congestion level 2*; otherwise, it is classified as *Congestion level 1*.
- **Node C:** At node C, the split is also based on b_2. If $b_2 \geq 0.01$, the path leads to node D; otherwise, it is classified as *Congestion level 3*.
- **Node D:** Node D is reached from node C when $b_2 \geq 0.01$ is true. At this terminal node, the observation is classified as *Congestion level 4*.

Matching these conditions with the options provided, we find that:

Answer: A: $A : b_1 \geq -0.16, \quad B : b_2 \geq 0.03, \quad C : b_2 \geq 0.01, \quad D : b_1 \geq -0.76$

Question 5. Spring 2019 question 22:

Given:

- Outer folds (K_1): 5

- Inner folds (K_2): 4
- Hyperparameter values for NN (n_h): 5 values
- Hyperparameter values for LR (λ): 5 values
- Training and testing times:
  - NN: Training = 20 ms, Testing = 5 ms
  - LR: Training = 8 ms, Testing = 1 ms

Calculations:

1. **Total number of trainings and tests for each model:**
   - **Inner Cross-Validation:**
     - For each outer fold:
       - For each hyperparameter value:
         - For each inner fold:
         - 1 training (on training data of inner fold)
         - 1 testing (on validation data of inner fold)
     - **Total trainings/tests per model in inner CV:**

       $$5 \text{ outer folds} \times 5 \text{ hyperparameters} \times 4 \text{ inner folds} = 100 \text{ trainings/tests}$$

   - **Outer Evaluation:**
     - For each outer fold:
       - 1 training (on training data of outer fold with optimal hyperparameter)
       - 1 testing (on test data of outer fold)
     - **Total trainings/tests per model in outer evaluation:**

       $$5 \text{ outer folds} \times 1 = 5 \text{ trainings/tests}$$

   - **Total trainings/tests per model:**

     $$100 + 5 = 105 \text{ trainings}$$

     $$100 + 5 = 105 \text{ tests}$$

2. **Total time for each model:**
   - **Neural Network:**
     - Total training time:

       $$105 \text{ trainings} \times 20 \text{ ms} = 2100 \text{ ms}$$

  - Total testing time:

$$105 \text{ tests} \times 5 \text{ ms} = 525 \text{ ms}$$

  - Total NN time:

$$2100 \text{ ms} + 525 \text{ ms} = 2625 \text{ ms}$$

- **Logistic Regression:**
  - Total training time:

$$105 \text{ trainings} \times 8 \text{ ms} = 840 \text{ ms}$$

  - Total testing time:

$$105 \text{ tests} \times 1 \text{ ms} = 105 \text{ ms}$$

  - Total LR time:

$$840 \text{ ms} + 105 \text{ ms} = 945 \text{ ms}$$

3. **Total time to compose the table:**

$$\text{Total time} = 2625 \text{ ms (NN)} + 945 \text{ ms (LR)} = 3570 \text{ ms}$$

Answer: C) 3570.0 ms

Question 6. Spring 2019 question 26:

To assign each observation to one of the four classes $y = 1, 2, 3, 4$, we use a multinomial regression model with a softmax transformation. For each observation $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, we compute three scores $\hat{y}_k$ for $k = 1, 2, 3$ as follows:

$$\hat{y}_k = \begin{bmatrix} 1 \\ b_1 \\ b_2 \end{bmatrix}^T w_k = 1 \cdot w_{k,0} + b_1 \cdot w_{k,1} + b_2 \cdot w_{k,2}$$

where the weights are:

$$w_1 = \begin{bmatrix} 1.2 \\ -2.1 \\ 3.2 \end{bmatrix}, \quad w_2 = \begin{bmatrix} 1.2 \\ -1.7 \\ 2.9 \end{bmatrix}, \quad w_3 = \begin{bmatrix} 1.3 \\ -1.1 \\ 2.2 \end{bmatrix}$$

After calculating $\hat{y}_1$, $\hat{y}_2$, and $\hat{y}_3$ for each observation, we use the softmax transformation to compute the probability of each class. The probability for class $y = 4$ is given by:

$$P(y = 4|\hat{y}) = \frac{1}{1 + \sum_{k'=1}^{3} e^{\hat{y}_{k'}}}$$

We are looking for the observation with the highest $P(y = 4|\hat{y})$.

Calculations for Each Observation

Observation A: $b = \begin{bmatrix} -1.4 \\ 2.6 \end{bmatrix}$

$$\hat{y}_1 = 1 \cdot 1.2 + (-1.4) \cdot (-2.1) + 2.6 \cdot 3.2 = 12.46$$

$$\hat{y}_2 = 1 \cdot 1.2 + (-1.4) \cdot (-1.7) + 2.6 \cdot 2.9 = 10.06$$

$$\hat{y}_3 = 1 \cdot 1.3 + (-1.4) \cdot (-1.1) + 2.6 \cdot 2.2 = 8.54$$

$$P(y = 4|\hat{y}) = \frac{1}{1 + e^{12.46} + e^{10.06} + e^{8.54}} \approx 3.03 \times 10^{-6}$$

Observation B: $b = \begin{bmatrix} -0.6 \\ -1.6 \end{bmatrix}$

$$\hat{y}_1 = 1 \cdot 1.2 + (-0.6) \cdot (-2.1) + (-1.6) \cdot 3.2 = -2.66$$

$$\hat{y}_2 = 1 \cdot 1.2 + (-0.6) \cdot (-1.7) + (-1.6) \cdot 2.9 = -2.42$$

$$\hat{y}_3 = 1 \cdot 1.3 + (-0.6) \cdot (-1.1) + (-1.6) \cdot 2.2 = -1.56$$

$$P(y = 4|\hat{y}) = \frac{1}{1 + e^{-2.66} + e^{-2.42} + e^{-1.56}} \approx 0.7305$$

Observation C: $b = \begin{bmatrix} 2.1 \\ 5.0 \end{bmatrix}$

$$\hat{y}_1 = 1 \cdot 1.2 + 2.1 \cdot (-2.1) + 5.0 \cdot 3.2 = 12.79$$

$$\hat{y}_2 = 1 \cdot 1.2 + 2.1 \cdot (-1.7) + 5.0 \cdot 2.9 = 12.13$$

$$\hat{y}_3 = 1 \cdot 1.3 + 2.1 \cdot (-1.1) + 5.0 \cdot 2.2 = 9.99$$

$$P(y = 4|\hat{y}) = \frac{1}{1 + e^{12.79} + e^{12.13} + e^{9.99}} \approx 1.77 \times 10^{-6}$$

Observation D: $b = \begin{bmatrix} 0.7 \\ 3.8 \end{bmatrix}$

$$\hat{y}_1 = 1 \cdot 1.2 + 0.7 \cdot (-2.1) + 3.8 \cdot 3.2 = 11.89$$

$$\hat{y}_2 = 1 \cdot 1.2 + 0.7 \cdot (-1.7) + 3.8 \cdot 2.9 = 11.03$$

$$\hat{y}_3 = 1 \cdot 1.3 + 0.7 \cdot (-1.1) + 3.8 \cdot 2.2 = 8.89$$

$$P(y = 4|\hat{y}) = \frac{1}{1 + e^{11.89} + e^{11.03} + e^{8.89}} \approx 4.66 \times 10^{-6}$$

Conclusion

Among all observations, Observation B has the highest probability for class $y = 4$ with $P(y = 4|\hat{y}) \approx 0.7305$. Therefore, **Observation B is assigned to class** $y = 4$.