California House Price Prediction

by: Rachel Chen, Yuzuka Rao

Introduction

dataset: 20641 housing data with these columns

longitude	float64
latitude	float64
housing_median_age	float64
total_rooms	float64
total_bedrooms	float64
population	float64
households	float64
median_income	float64
median_house_value	float64
ocean_proximity	object

- Grouped these housing prices into 16 categories
- Classification problem

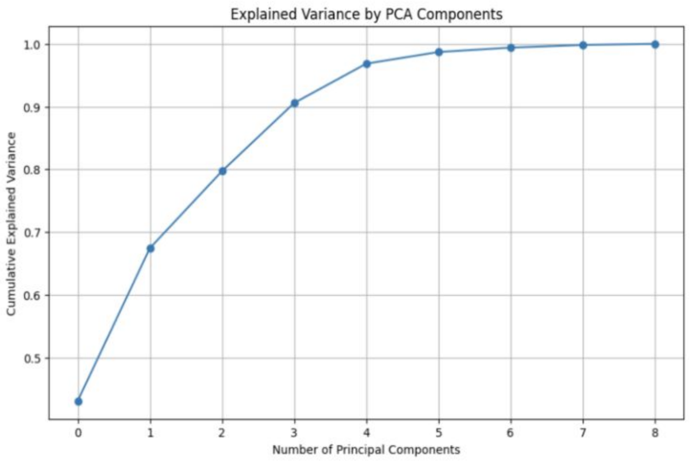
Unsupervised Analysis

- Clean null values in “total_bedrooms” column
- Do ordinal encoding for “ocean_proximity” data

```
ocean_proximity_mapping = {
    '<1H OCEAN': 3,
    'NEAR OCEAN': 2,
    'NEAR BAY': 1,
    'INLAND': 0
}
```

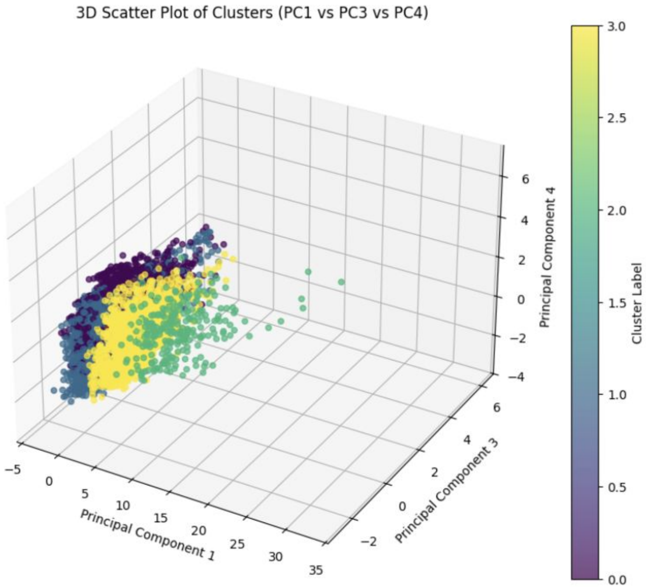
Unsupervised Analysis

PCA



- 5 components covers 96.84% variance

Clustering



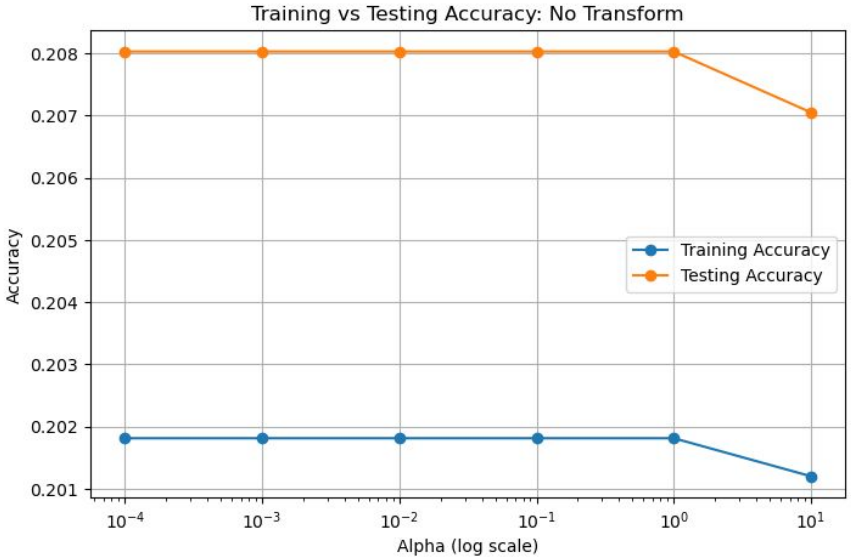
- 4 clusters
- distance to each cluster is added as feature

Supervised analysis

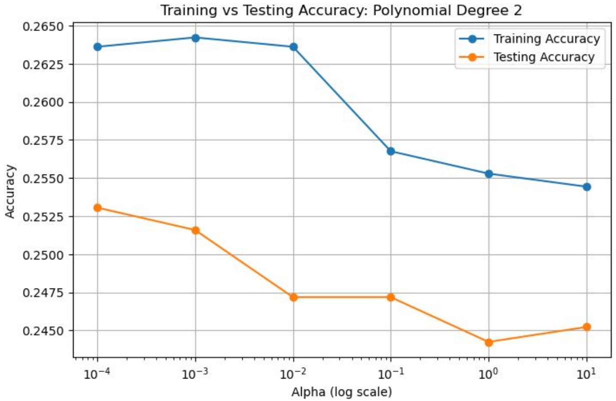
- Neural Networks
  - Regularization 1-6
  - different architectures and layers
- Svm
  - At least 3 features transformation (polynomial, PCA, or radial-basis function kernel)
- Linear Regression

Linear Regression

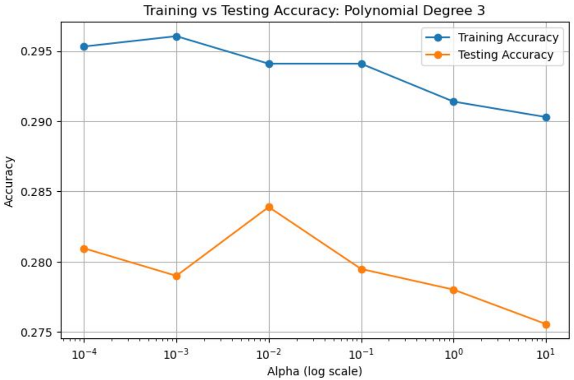
- Initial idea: use Linear Regression, predict a value, then round to the nearest price range class
- Adding Ridge Regularization: different alpha values from 0.001 to 100
- Train Accuracy: ~20.2%
- Test Accuracy: ~20.8%
- High MSE and low precision/recall → underfitting.

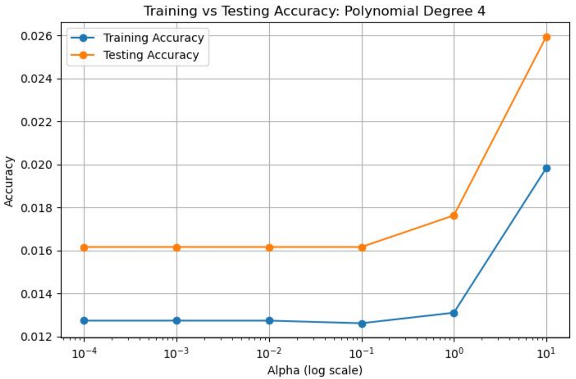


Polynomial Transformatic



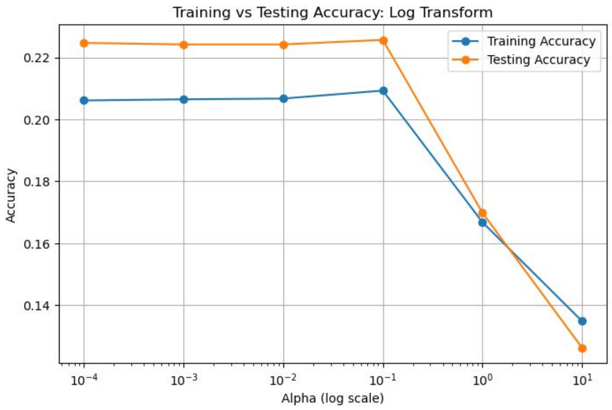
- Best result came from:
  - Polynomial degree 3 + low regularization
  - Train Accuracy: ~29.5%
  - Test Accuracy: ~28.3%
- Degree 3: mild improvement, starts to overfit.
- Degree 4: unstable performance, catastrophic overfitting.

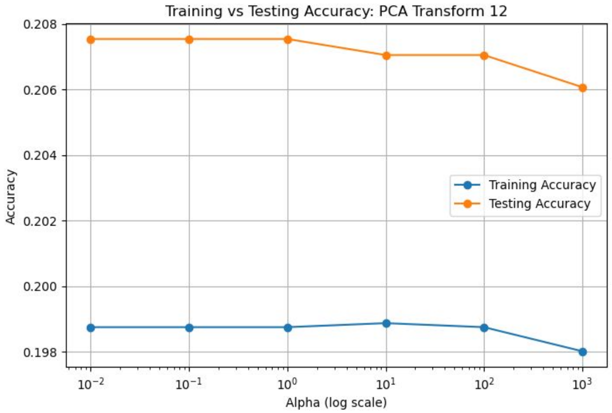




Log and PCA Transformati

- Log Transformation: data skew
    - Train Accuracy: ~20.6%
    - Test Accuracy: ~22.5%
    - Increase regularization make performance deteriorated
- PCA: reduce dimensionality
    - Train Accuracy: ~19.9%
    - Test Accuracy: ~20.7%





Conclusion & Improvement

Result:

- Linear regression, even with transformations, can’t model the problem well

Improvement:

- revert the data back to continuous data
- Better performance, but still generalize poor

No transformation

```
Train Accuracy: 0.2192
Test Accuracy: 0.2173
Train Precision: 0.2746
Test Precision: 0.2724
Train Recall: 0.2192
Test Recall: 0.2173
```

Polynomial Degree 2

```
Train Accuracy: 0.2822
Test Accuracy: 0.2697
Train Precision: 0.3256
Test Precision: 0.3122
Train Recall: 0.2822
Test Recall: 0.2697
```

Polynomial Degree 3

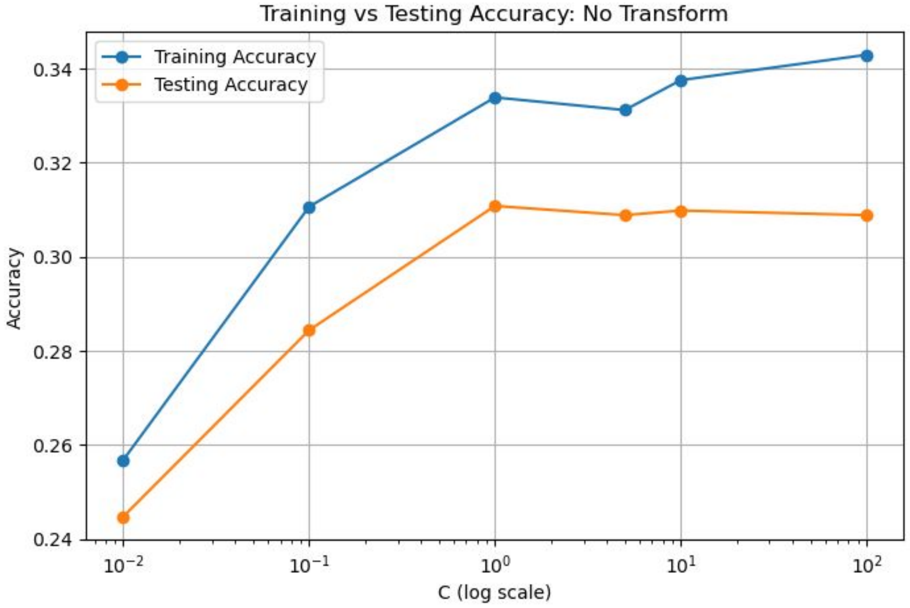
```
Train Accuracy: 0.3076
Test Accuracy: 0.2834
Train Precision: 0.3441
Test Precision: 0.3194
Train Recall: 0.3076
Test Recall: 0.2834
```

Polynomial Degree 4

```
Train Accuracy: 0.3140
Test Accuracy: 0.2805
Train Precision: 0.3522
Test Precision: 0.3124
Train Recall: 0.3140
Test Recall: 0.2805
```

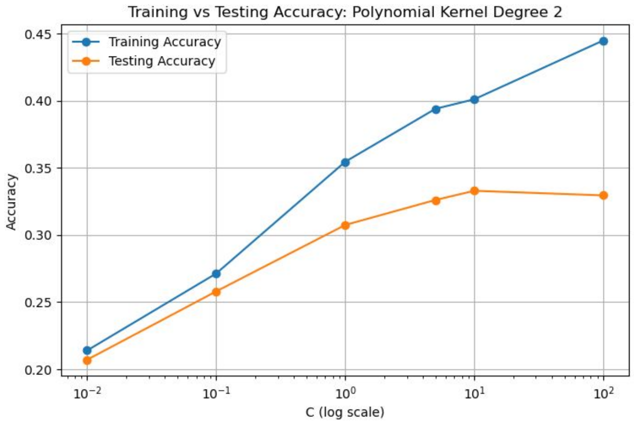
SVM

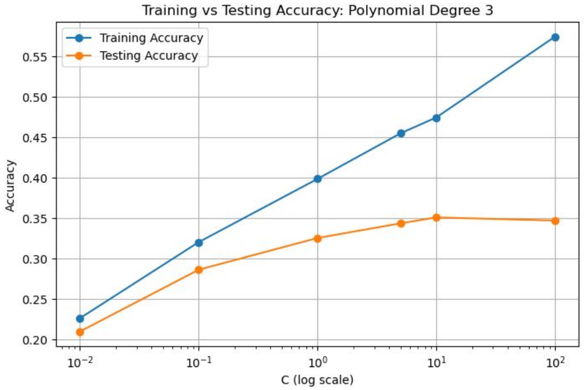
- Linear kernel with regularization
  - Train Accuracy: ~33%
  - Test Accuracy: ~31%
  - Better than linear regression, no overfitting

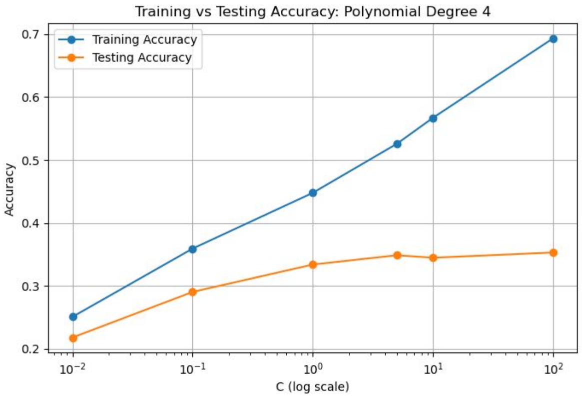


Polynomial Transformat

- Polynomial Degree 2
  - better training fit, mild overfit
  - Train Accuracy: ~44%
  - Test Accuracy: ~33%
- Polynomial Degree 3–4:
  - Train Accuracy up to 50%+
  - Test Accuracy stagnated lead to overfit

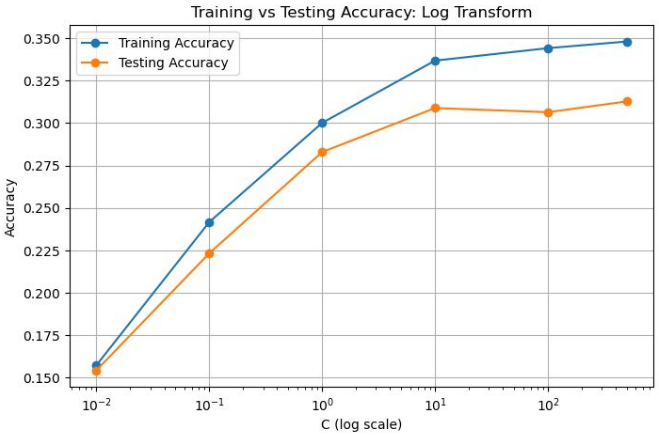


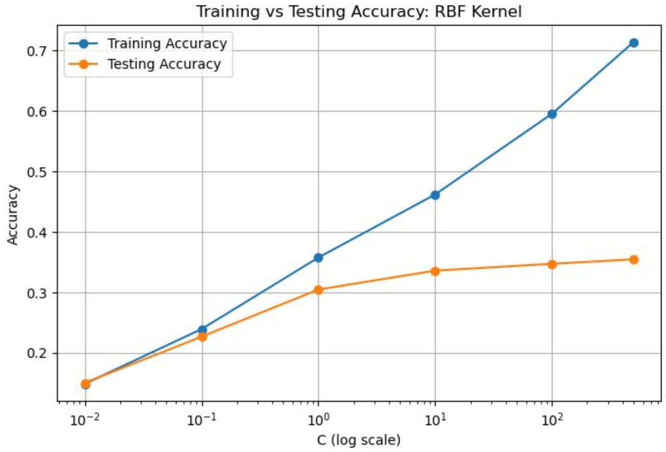




Log Transformation and RBF

- Log Transformation:
  - Barely impact
  - But no underfit compare to linear regression model
  - Train Accuracy: ~34.8%
  - Test Accuracy: ~31.3%
- Best result: RBF Kernel
  - Train Accuracy: ~71%
  - Test Accuracy: 35.4%

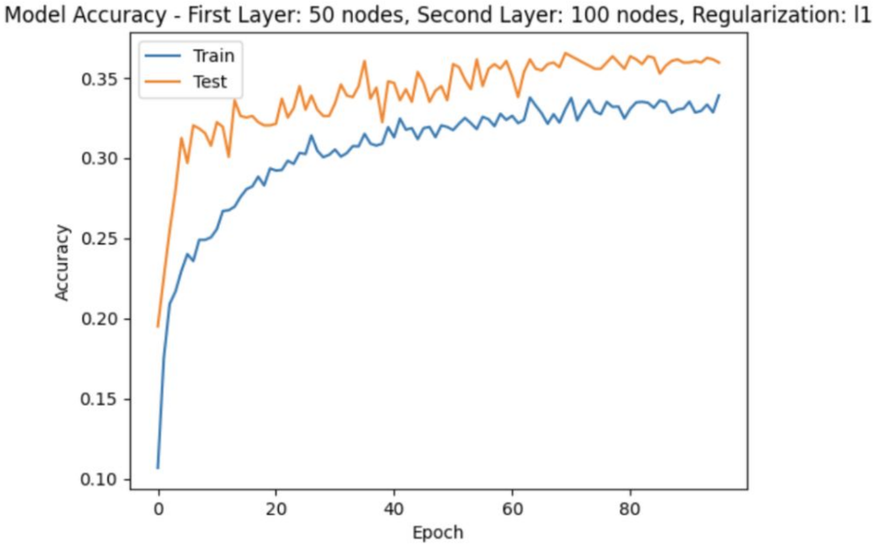




Neural Network

- first layer node: [50,200,500]
- second layer node: [50,100]
- Regularization type: [‘L1’, ‘L2’]

Underfitting



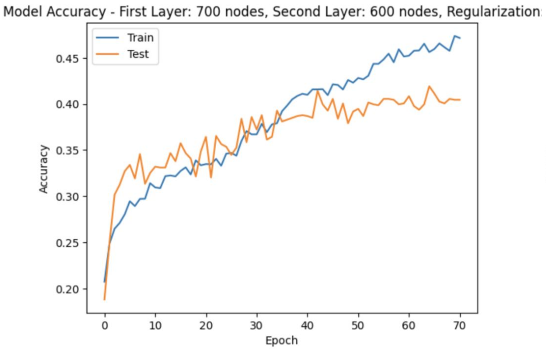
Neural Network

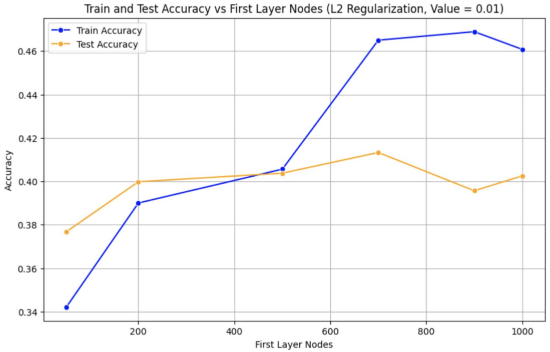
- Change first layer node to: [700, 900, 1000]
- second layer node: [600,800]
- Regularization type: ‘L2’

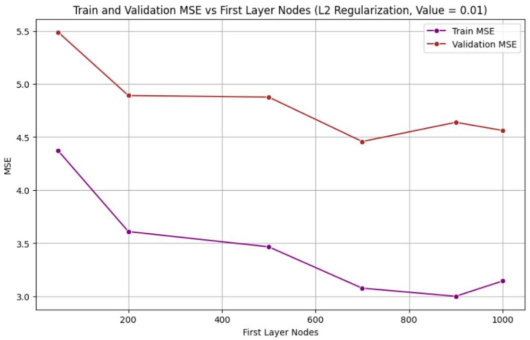
↓

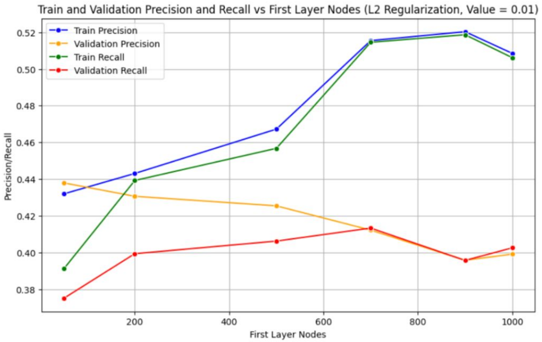
Better performance

Accuracy 41.63%



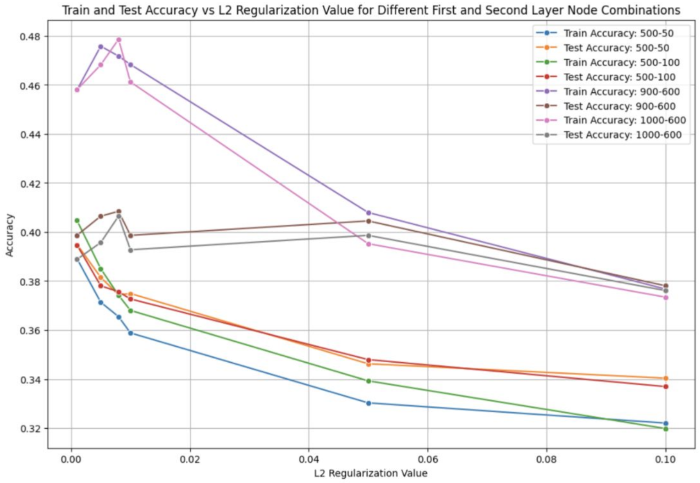






Neural Network

Applying L2 regularization:



Applying polynomial degree 2 transformation:

	Average accuracy	Average Val precision
No transformation	36.30%	41.60%
Poly degree 2 transformation	36.39%	41.39%
	Average val recall	Average Val MSE
No transformation	37.02%	5.47
Poly degree 2 transformation	36.79%	5.49

Conclusion

Overall, SVM model is able to capture complex nonlinear relationships. It improves a lot compared to the linear regression model.

SVM Table

Transformation	C	Train Accuracy	Test Accuracy	Precision	Recall
No Transform	0.01	0.25679315	0.24473813	0.19147568	0.24473813
No Transform	0.1	0.31064872	0.28438571	0.21199272	0.28438571
No Transform	1	0.33390453	0.31081743	0.26243	0.31081743
No Transform	5	0.33121175	0.30885952	0.26244321	0.30885952
No Transform	10	0.3375765	0.30983847	0.28065783	0.30983847
No Transform	100	0.34296206	0.30885952	0.27796924	0.30885952

Linear regression table

Transformation	Alpha	Train Accuracy	Test Accuracy	Precision	Recall	Train MSE	Test MSE
No Transform	0.001	0.20181128	0.20802741	0.21924137	0.20802741	5.25050684	5.44496052
No Transform	0.01	0.20181128	0.20802741	0.21924137	0.20802741	5.25050684	5.44496067
No Transform	0.1	0.20181128	0.20802741	0.21924137	0.20802741	5.25050684	5.44496212
No Transform	1	0.20181128	0.20802741	0.21924137	0.20802741	5.25050706	5.4449768
No Transform	10	0.20119936	0.20704846	0.21857622	0.20704846	5.25052879	5.44514074
No Transform	100	0.19960837	0.20606951	0.21750721	0.20606951	5.25235748	5.44814359

Conclusion

Our neural networks model has:

Model Training: Trained 144 models with varying hyperparameters:

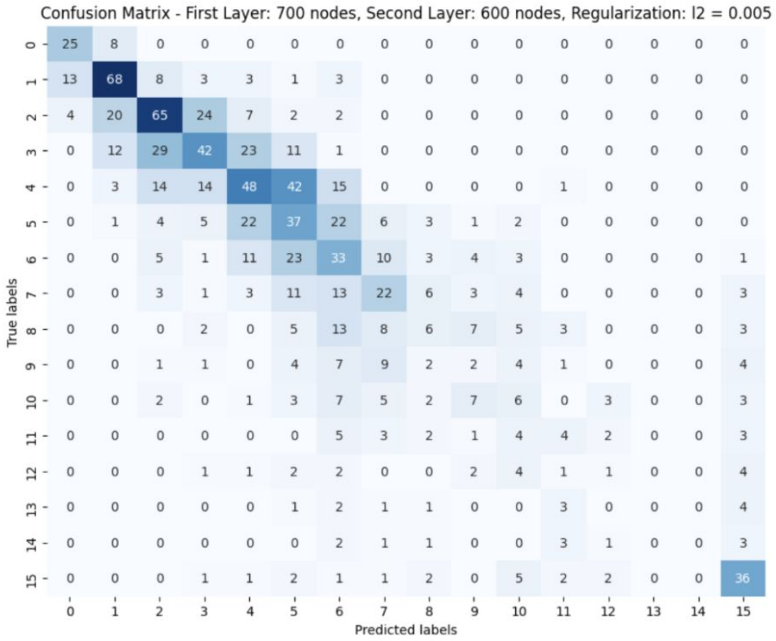
- First layer nodes: 50, 200, 500
- Second layer nodes: 50, 100
- Regularization: L1, L2 (with values: 0.001, 0.005, 0.008, 0.01, 0.05, 0.1)
- Polynomial transformation (degree 2 or not)

Best Performance: Achieved **41.33% accuracy**, **40.74% validation recall**, and **42.22% validation precision** with 200 first layer nodes, 100 second layer nodes, and L2 regularization (0.01).

Underfitting Issue: test accuracy is higher than train accuracy

Further Optimization: Trained an additional 36 models with larger node configurations (700, 900, 1000 nodes for first layer and 600, 800 nodes for second layer).

Improved Performance: The highest test accuracy reached **41.63%**.



Future Work

- Use SMOTE to make data to be more evenly distributed, not left-skewed
- Training neural network with powerful computer to find the optimized solution

Thank you for listening!