Ridge regression

- **Idea**: In addition to just minimizing mean squared error, what if we could **also** try and prevent large parameter values?

  Maybe this would lead to less overfitting!

- **Regularization** is the act of adding a penalty on the norm of the parameter vector, $\vec{w}$, to the objective function.

$$R_{\text{ridge}}(\vec{w}) = \underbrace{\frac{1}{n}\|\vec{y} - X\vec{w}\|^2}_{\text{MSE}} + \underbrace{\lambda \sum_{j=1}^{d} w_j^2}_{\text{regularization penalty}}$$

$= \frac{1}{n} \sum_{i=1}^{n} (y_i - H(\vec{x}_i))^2$

↑ regularization hyperparameter.

- **Idea**: In addition to just minimizing mean squared error, what if we could **also** try and prevent large parameter values?

  Maybe this would lead to less overfitting!

- **Regularization** is the act of adding a penalty on the norm of the parameter vector, $\vec{w}$, to the objective function.

$$R_{\text{ridge}}(\vec{w}) = \frac{1}{n}\|\vec{y} - X\vec{w}\|^2 + \lambda \underbrace{\sum_{j=1}^{d} w_j^2}_{\text{regularization penalty}}$$

$= \|\vec{w}\|^2$, but without penalizing w_0^2

- Linear regression with L_2 regularization – as shown above – is called **ridge regression**.

  You'll explore the reason why in Homework 9!

$$\|\vec{w}\|^2 = w_0^2 + w_1^2 + \cdots + w_d^2$$

$$R_{\text{ridge}}(\vec{w}) = \frac{1}{n}\|\vec{y} - X\vec{w}\|^2 + \underbrace{\lambda \sum_{j=1}^{d} w_j^2}_{\text{regularization penalty}}$$

- Linear regression with L_2 regularization – as shown above – is called **ridge regression**.
  You'll explore the reason why in Homework 9!

- Intuition: Instead of just minimizing mean squared error, we balance minimizing mean squared error and a penalty on the size of the fit coefficients, $w_1^*, w_2^*, ..., w_d^*$.
  We don't regularize the intercept term!

- λ is a **hyperparameter**, which we choose through cross-validation.

- The $\vec{w}^*_{\text{ridge}}$ that minimizes $R_{\text{ridge}}(\vec{w})$ is not necessarily the same as $\vec{w}^*_{\text{OLS}}$, which minimizes $R_{\text{sq}}(\vec{w})$!

$= (X^T X)^{-1} X^T \vec{y}$

$H(\vec{x}_i) = w_0 + w_1 x_i^{(1)} + w_2 x_i^{(2)} + \ldots + w_d x_i^{(d)}$

Activity

The objective function we minimize to find $\vec{w}^*_{\text{ridge}}$ in **ridge regression** is:

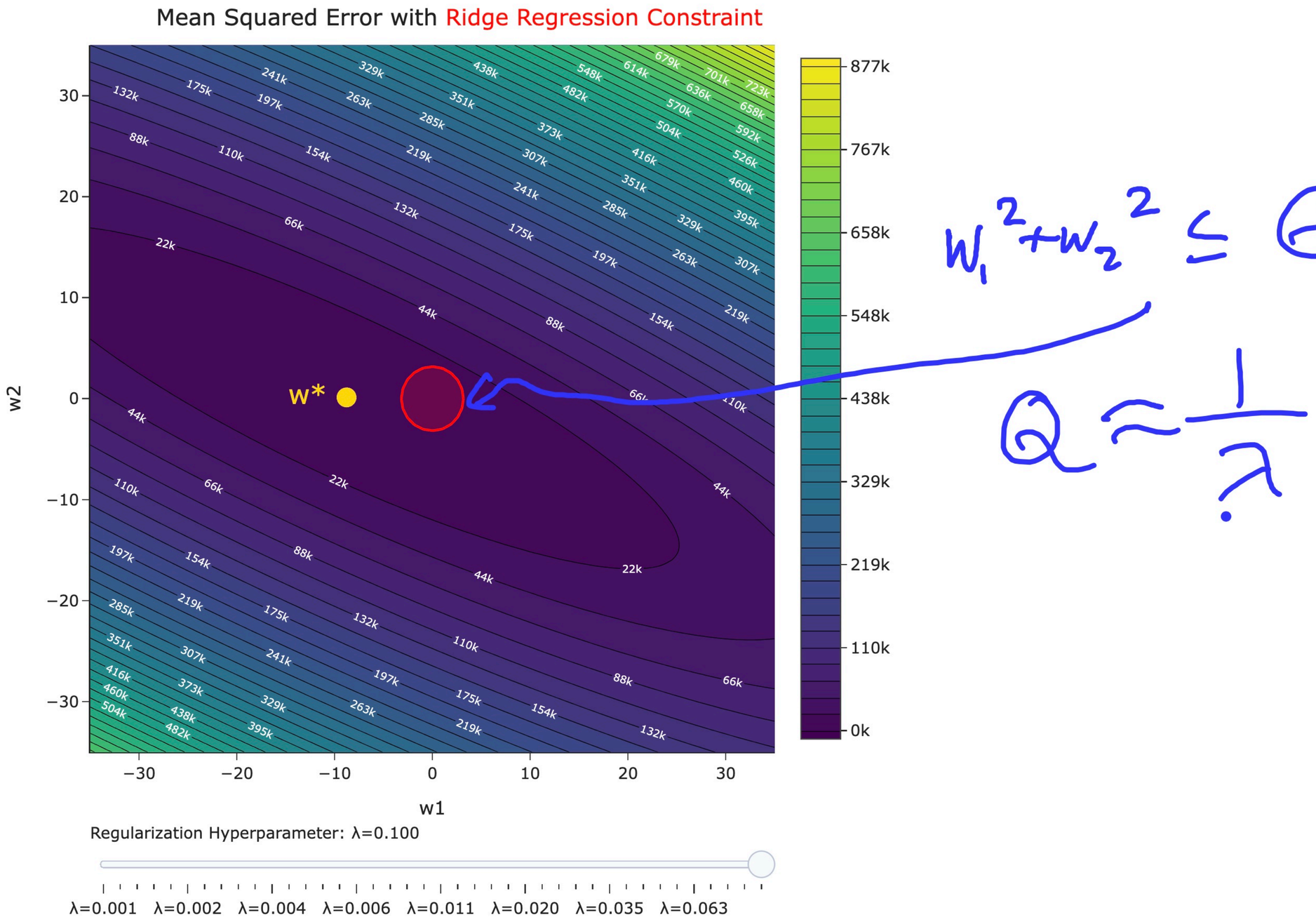
$$R_{\text{ridge}}(\vec{w}) = \frac{1}{n}\|\vec{y} - X\vec{w}\|^2 + \lambda \sum_{j=1}^{d} w_j^2$$

λ is a **hyperparameter**, which we choose through cross-validation. Discuss the following points with those near you:

- What if we pick $\lambda = 0$ – what is $\vec{w}^*_{\text{ridge}}$ then?
- What happens to $\vec{w}^*_{\text{ridge}}$ as $\lambda \to \infty$?
- Can λ be negative? no!

as $\lambda \to \infty$, $\vec{w}^*_{\text{ridge}} \to$ [mean(y), 0, 0, ..., 0] (d 0s)

In [9]:
```
1 util.show_ridge_contour()
```



Using `GridSearchCV` to choose λ

$10^{-2}, 10^{-1}, \cdots, 10^{14}$ try large range.

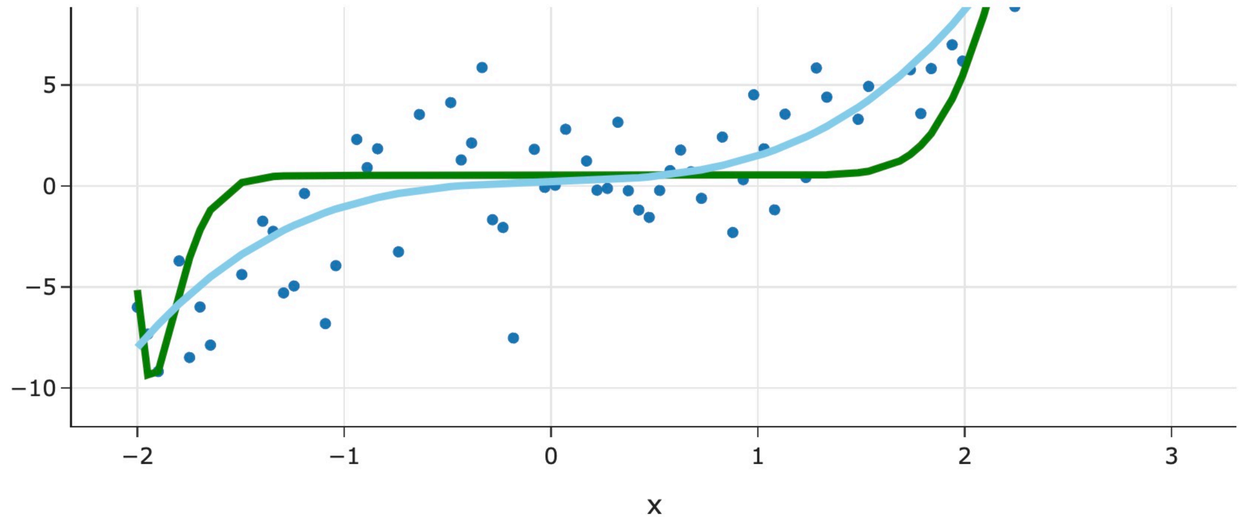
- In general, we won't just arbitrarily choose a value of λ.

- Instead, we'll perform k-fold cross-validation to choose the λ that leads to predictions that work best on unseen test data.

  The value of λ depends on the specific dataset and model you've chosen; there's no universally "best" λ.

In [ ]:

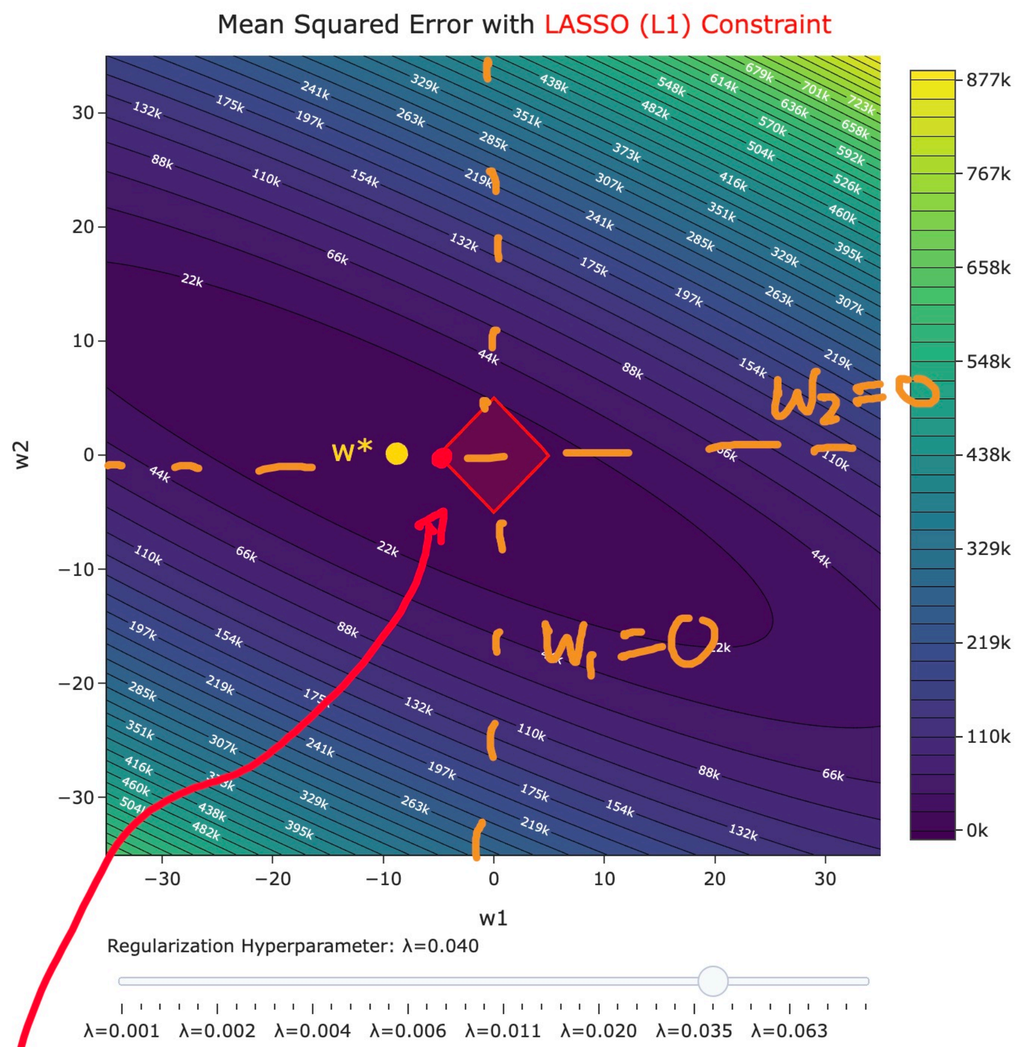
```
hyperparams = {
    'ridge__alpha': 10.0 ** np.arange(-2, 15) # Try 0.01, 0.1, 1, 10, 100, 1000, ...
}
model_regularized = GridSearchCV(
    estimator=make_pipeline(PolynomialFeatures(25, include_bias=False), Ridge()),
    param_grid=hyperparams,
    scoring='neg_mean_squared_error'
)
model_regularized.fit(X_train, y_train)
```



In [34]:
```
1 display(HTML(results_df_str))
```

	Unregularized (Degree 25)	Regularized (Degree 25) Used cross-validation to choose λ	Regularized (Degree 3) Used cross-validation to choose λ and degree
training MSE	4.72	10.33	7.11
average validation MSE (across all folds)	NaN	17.60	7.40
test MSE	14.21	17.17	10.52

In [44]:
```
util.show_lasso_contour()
```



Corners of are where coefficients $= 0$!

$\hat{w}_{LASSO}$ snaps to corners of constraint set!

Aside: average validation error vs. λ

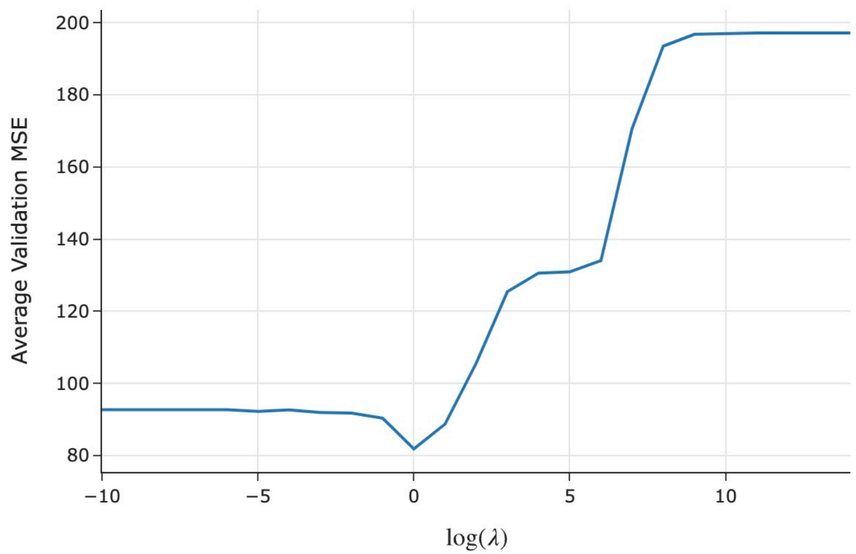
next class!.

- How did the average validation MSE change with λ?

  Here, large values of λ mean **less complex models**, not more complex.

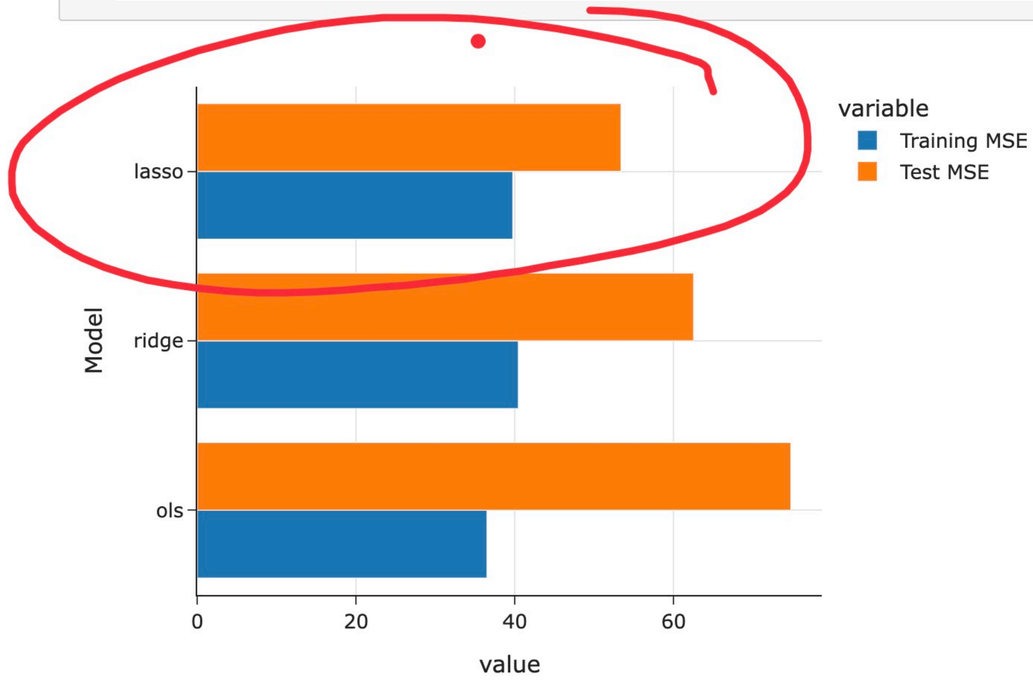
In [54]:

```
(
    pd.Series(-commute_model_ridge.cv_results_['mean_test_score'],
              index=np.log10(lambdas))
    .to_frame()
    .reset_index()
    .plot(kind='line', x='index', y=0)
    .update_layout(xaxis_title='$\log(\lambda)$', yaxis_title='Average Validation MSE')
)
```



feature	ols	ridge	lasso
intercept	460.31	214.15	2.54e+02
polynomialfeatures__departure_hour	-94.79	-0.71	-2.10e+01
polynomialfeatures__departure_hour^2	6.80	-4.63	-1.70e+00
polynomialfeatures__departure_hour^3	-0.14	0.31	1.81e-01
onehotencoder__day_Mon	-0.61	-5.74	-2.70e+00
onehotencoder__day_Thu	13.30	6.04	9.00e+00
onehotencoder__day_Tue	11.19	5.52	8.68e+00
onehotencoder__day_Wed	5.73	-0.46	0.00e+00
onehotencoder__month_December	8.90	2.82	4.06e+00
onehotencoder__month_February	-5.33	-7.14	-5.81e+00
onehotencoder__month_January	1.93	0.39	0.00e+00
onehotencoder__month_July	2.46	0.44	0.00e+00
onehotencoder__month_June	6.28	4.45	5.14e+00
onehotencoder__month_March	-0.76	-1.70	-8.17e-01
onehotencoder__month_May	9.36	4.95	5.57e+00
onehotencoder__month_November	1.40	-1.81	-0.00e+00

```
}).set_index('Model')
df.plot(kind='barh', barmode='group')
```



- The best-fitting LASSO model seems to have a lower training and testing MSE than the best-fitting ridge model.

- But, in general, sometimes LASSO performs better on unseen data, and sometimes ridge does. Cross-validate!

  Sometimes, machine learning practitioners say "there's no free lunch" – there's no universal always-best technique to use to make predictions, it always depends on the specific data you have.